WALKOWIAK

ADVENTURES



UND WIE MAN SIE AUF DEM

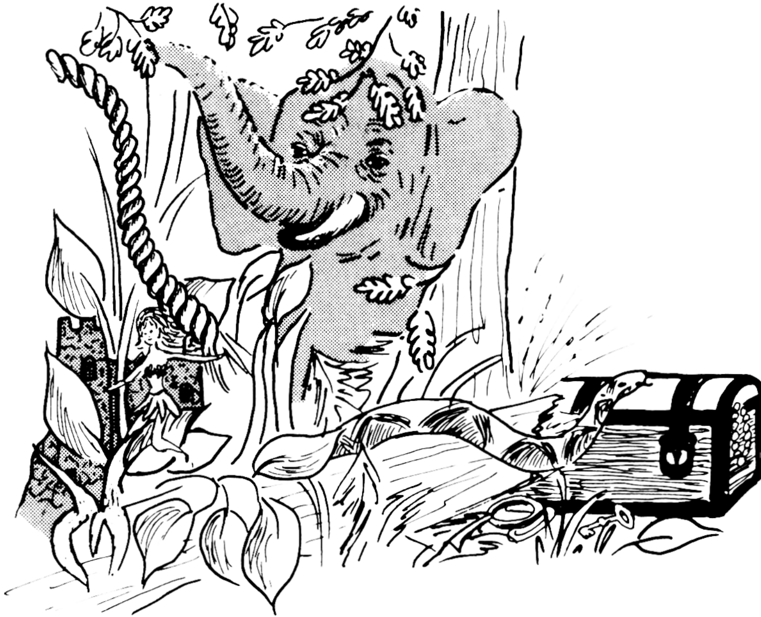
CPC 464

PROGRAMMIERT

EIN DATA BECKER BUCH

WALKOWIAK

ADVENTURES



UND WIE MAN SIE AUF DEM

CPC 464

PROGRAMMIERT

EIN DATA BECKER BUCH

ISBN 3-89011-088-6

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von dem Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

INHALTSVERZEICHNIS

VORWORT

*"**ABENTEUERSPIELE (ADVENTURE GAMES)**, für einige das Höchste, für andere schlicht langweilig. Ein Computerspiel mit viel Text, bei dem der Spieler eingeladen ist, an einer Reihe von Zufallsereignissen teilzunehmen und sich in einem Irrgarten oder Labyrinth zurechtzufinden."*

Diese Definition findet der interessierte Leser im Anhang des Handbuches zum Schneider CPC 464, und sollte er ihr Glauben schenken und das Thema Adventures nun mit einem 'Ach herrje' abtun, wird er auf die interessanteste Art, sich auf spielerische Art und Weise mit seinem Computer zu befassen, verzichten müssen.

Und dies wäre wirklich sehr schade, denn warum sollten Sie auf etwas verzichten, woran Millionen Computerfreunde in aller Welt Spaß haben ?

Denn wie wollen Sie es sich sonst erklären, wenn Sie Kollegen in der Schule oder am Arbeitsplatz sehen, die ihre Pause mit einer Diskussion darüber verbringen, wie der Held die lebensbedrohliche Situation zum Guten wenden kann ?
Warum lassen sich in jeder einschlägigen Fachzeitschrift Artikel zum Thema Adventure finden ?
Und warum steigen auch Riesen der Unterhaltungsbranche, wie WARNER BROS. oder LUCASFILM (geschützte Namen), in den Markt der Adventurespiele ein und begnügen sich nicht mit ihren auf Celluloid dargestellten Welten ?

Warum, wenn die Abenteuerspiele wirklich so wenig zu bieten haben ?

Diese Frage werden Sie sich und jedem anderen beantworten können, wenn Sie sich mit diesem Buch befaßt haben.

Denn mit '*Adventures - und wie man sie auf dem CPC 464 programmiert*' halten Sie den Schlüssel zur Welt der Adventurespiele in der Hand, zu einer Welt, die jeder betreten sollte, der über dem Ernst des Lebens noch nicht den Frohsinn der spielend verbrachten Tage seiner Kindheit vergessen hat.

Sie werden Abenteuer in grauer Vorzeit und im All erleben, und dabei werden Sie feststellen, daß Zufälle im Adventure höchst selten dem Zufall überlassen bleiben; daß Labyrinthe und Irrgärten immer nur Beiwerk sind, um den Spieler zusätzlich zu verwirren, und daß Abenteuerspiele keinesfalls immer nur reine Textspiele sein müssen - schon gar nicht, wenn sie auf einem Computer wie dem CPC 464 laufen.

Schritt für Schritt werden Sie gemeinsam mit mir eine Konzeption zur Realisierung von Abenteuerspielen entwickeln, wobei wir deren optische wie auch spieltechnische Erscheinungsform nach professionell vertriebenen Adventurespielen ausrichten werden.

Selbst wenn Sie sich nicht zu den Programmierprofis zählen, kurze Exkurse, zur Kennzeichnung *kursiv* gedruckt, werden Sie in die zum Verständnis der Programme wichtigen Grundlagen einführen, so daß Sie auch als Anfänger mit dem Inhalt dieses Buches ohne Schwierigkeiten zurechtkommen werden.

Dabei hilft Ihnen auch der Aufbau der Programme; nach jedem Schritt werden Sie die Richtigkeit der Überlegungen sogleich in der Praxis erproben können.

So programmieren wir zunächst die Handlungswelt eines Adventures, dann werden wir uns überlegen, wie dem Spieler die gezielte Fortbewegung möglicht gemacht werden kann, und wie wir ihm Gegenstände zur Hand geben können, mit denen er arbeiten kann.

Ein weiteres Kapitel wird Sie in die besonderen Geheimnisse der Adventureprogrammierung einweihen; es wird Ihnen zusätzliche Vorschläge machen, wie Sie einem Spieler Ihrer Adventures das Leben erschweren können.

Darüber hinaus finden Sie in diesem Kapitel genaue Anweisungen, wie Sie jedes erstellte Textadventure zu einem Grafikadventure ausbauen können.

Und damit die Gestaltung der Grafiken nicht allzu arbeitsaufwendig wird, erhalten Sie auch das Listing eines Grafik Programmers, der die erforderlichen Unterprogramme für Sie erzeugen wird.

Aber es bleibt nicht allein bei diesem Hilfsprogramm, sondern mit dem Listing ´Venturefix´ stellen wir Ihnen einen Adventuregenerator zur Verfügung, der es Ihnen, selbst wenn Sie Ihren CPC erst heute erstanden haben sollten und somit über keinerlei Programmierkenntnisse verfügen, erlaubt, voll funktionsfähige Adventureprogramme zu erzeugen.
Noch komfortabler wird die Sache mit unserem Adventuresystem, bestehend aus einem Adventureeditor mit zugehörigem Interpreter, nun können Sie jede Idee und jeden Gedankengang sofort im Spiel überprüfen und korrigieren.

Sollten Sie es allerdings vorziehen, zunächst erst einige Adventures zu spielen, so schlagen Sie bitte das letzte Kapitel auf, denn hier finden Sie die Textadventures *Goldrausch* und *Das verzauberte Schloß*, wie auch ein Listing für das Grafikadventure *Space Mission*.

Ich nehme an, daß Sie sich nun lieber mit dem eigentlichen Thema dieses Buches statt mit einer langen Vorrede befassen wollen, und so bleibt mir im Moment nur noch übrig, mich

für den Kauf dieses Buches recht herzlich bei Ihnen zu bedanken.

Darüber hinaus gilt mein besonderer Dank Herrn Dr. Achim Becker, der die Herausgabe dieses Buches möglich gemacht hat, wie auch Frau Alicia Clees und Herrn Claus Wagner, die mich bei der redaktionellen Arbeit auf unermüdliche Weise unterstützt haben.

Recklinghausen,
Dezember 1984 Jörg Walkowiak

1. KAPITEL
- EINFÜHRUNG -

ABENTEUER, sind außergewöhnliche Erlebnisse eines Menschen, die durch extreme Situationen und Gefahren gekennzeichnet sind und sich unauslöschlich der Erinnerung des *Abenteurers* einprägen.
Sie üben einen großen Einfluß auf die Psyche des Menschen aus und faszinieren bereits seit undenklichen Zeiten auch an den Geschehnissen unbeteiligte Personen.
Diese Eigenschaft machte sich die Dichtung zunutze, und der Lauf der Zeit führte zur Entwicklung diverser literarischer Spielarten.
Waren es zunächst Minnesänger, die dem Volke oder auch gekrönten Häuptern von den Taten ferner Helden berichteten, so erwuchsen diese Spielmannsdichtungen im 12. Jahrhundert zu großangelegten Epen wie ´Lanzelot´ oder Wolfram von Eschenbachs ´Parzifal´.
´Don Quijote´ und ´Münchhausen´ begeisterten mit ihren Erlebnissen ebenfalls Tausende.
Nach dieser Zeit des Schelmenromans machten sich die Reiseerzählungen (Defoe, Cooper, Karl May) das Thema Abenteuer zunutze bis die Entwicklung mit Kriminal- und Science Fiction Romanen ihren Abschluß fand.

ABENTEUER HEUTE

Wer kennt sie nicht, die auf der vorangegangenen Seite, welche einem Lexikon unserer Tage entstammen könnte, erwähnten Abenteuerromane; wer hat sich in seiner Kindheit nicht von den Erlebnissen Old Shatterhands oder Robinson Crusoes mitreißen lassen ? - Wer hat sich nicht mit den Hauptpersonen identifiziert und sich vorgestellt, wie man sich wohl selbst in der gleichen Situation verhalten hätte?

Doch vielleicht gehören Sie tatsächlich zu den wenigen, die aus Gründen der Bequemlichkeit niemals ein solches Buch gelesen haben. Warum auch, werden Sie dann fragen, ich kann es doch viel bequemer haben !
Don Quijote habe ich im Fernsehen gesehen, Lederstrumpf und Karl May dank der Programmdirektoren unserer Sender sogar öfter, warum sollte ich also meine Freizeit in irgendeiner Weise aktiv gestalten, wenn es auch anders geht, und außerdem: Lesen ist doch schon längst nicht mehr zeitgemäß.

Sicher, mit dem letzten Argument hätten Sie vermutlich recht, denn waren die Kinderstuben früher mit Puppen, Baukästen, Eisenbahnen oder Kinderbüchern gefüllt, so sind es heute ferngesteuerte Modelle, Experimentierkästen, Videorecorder, Telespiele und Heimcomputer. Unaufhaltsam ist die Technik auch hier auf dem Vormarsch, was bei richtiger Anwendung jedoch durchaus zu begrüßen ist.
Es mag wohl Computerbenutzer geben, die als einzige Anwendung realistische Raumschlachten und diverse andere Schieß- und Actionspiele kennen, doch irgendwann werden auch diese Anwender bemerken, daß der Spielablauf immer der gleiche ist. Denken ist nicht gefragt - nur der Bewegungsrhythmus der Steuertasten ändert sich von Spiel zu Spiel ein wenig.
Also unterzieht man den Softwaremarkt einer neuerlichen Prüfung - und stellt fest, daß die angloamerikanischen Computerbesitzer viel besser bedient werden; denn neben den

bekannten Arcade - Games haben sich in Amerika und England längst die Adventurespiele etabliert.
Adventures, zu deutsch Abenteuer, stellen neben Strategiespielen wie Schach oder Othello sicherlich die intelligenteste Anwendung eines Computers zu Spielzwecken dar.

Diese Programme ermöglichen es jedem Computerbesitzer , ohne große Risiken und Kosten die interessantesten Abenteuer zu erleben, wobei selbst das Unmögliche zum Normalzustand werden kann.
Stellen Sie sich einen Androiden vor, den Sie mit knappen Anweisungen durch alle Gefahren steuern, dem Sie Anweisungen geben, die er dann an Ihrer Stelle ausführt und die über Sieg oder Niederlage entscheiden.
Befehlen Sie ihm, daß er in eine bestimmte Richtung gehen soll, und er wird Ihnen daraufhin mitteilen, was er an seinem neuen Aufenthaltsort sieht, oder er wird Sie darüber aufklären, daß leider kein Weg in der gewünschten Richtung existiert.
Lassen Sie ihn Dinge, die er findet, näher untersuchen und befehlen Sie ihm, wenn Sie es für sinnvoll halten, diese Gegenstände zu nehmen.
Schließlich glauben Sie, eine im Hinblick auf das Spielziel sinnvolle Einsatzmöglichkeit für ein jedes dieser Objekte gefunden zu haben und lassen den Androiden entsprechend handeln; genießen Sie Ihren Triumph oder stellen Sie an Ihrem sich langsam entwickelnden Groll fest, wie sehr dieses Spiel Sie gepackt und der alltäglichen Welt entrissen hat.

Haben Sie ein Adventure in den Speicher Ihres Computers geladen, stehen Ihnen für Ihr Spiel nicht nur einige farbenfrohe Screens zur Verfügung, sondern Sie agieren in einer komplett ausstaffierten Welt, einer Welt deren Geheimnisse und Rätsel nur darauf warten, von Ihnen entdeckt und gelöst zu werden.

Dabei wird ein gutes Adventure Sie wie im richtigen Leben agieren lassen, das heißt, Aktion und Reaktion stehen in gewohntem Zusammenhang, Lebenserfahrungen erweisen sich auch im Computerspiel als nützlich.

Sie werden entdecken, daß selbst eine alltägliche Handlung wie das Öffnen einer Tür Schwierigkeiten bereiten kann, die sich mit Ausgabe einer Meldung wie *Ich habe den passenden Schlüssel nicht !* ankündigen.
Selbst Magie kann erforderlich sein, um als Antwort *O.K. - Die Tür ist auf.* zu erhalten, denn nur die Phantasie des Programmierers setzt die Grenzen zwischen Möglichem und Unmöglichem.

Andererseits kann das Adventurespiel aber auch als Lehrmittel eingesetzt werden. So ist beispielsweise ein Adventure, ausgestattet mit den Daten unseres Sonnensystemes denkbar, welches es uns gestattet, den Weltraum ganz bequem von unserem Lieblingssessel aus zu erforschen.

Welche Methode könnte besser geeignet sein, um Kindern oder Wißbegierigen gleich welchen Alters Allgemeinwissen oder auch spezielle Informationen nahezubringen ?

Wie auch immer, ob ein Adventure nun als Lehrprogramm oder nur zur Unterhaltung geschrieben wurde, ein gutes Reaktionsvermögen ist bei weitem nicht mehr ausreichend, um ein Spiel zu bewältigen.
Sie werden schon einen scharfen Verstand und logisches Denkvermögen brauchen, um die zahlreichen Gefahren im Leben eines Abenteurers meistern zu können:

Wie wollen Sie einen Fisch fangen, damit Sie nicht verhungern müssen, wenn Ihnen nur die bloßen Hände zur Verfügung stehen ?
Wie können Sie den hungrigen Bären davon abhalten, Sie als Zwischenmahlzeit zu betrachten?
Wie wollen Sie einen reißenden Fluß überqueren, der zudem noch von urweltlichen Raubtieren bewohnt wird ?

Mit einigem Nachdenken lassen sich all diese Probleme lösen, Sie müssen nur die richtigen Ideen und entsprechend viel Phantasie haben:

Ihre Aufgabe besteht darin, in ein bestimmtes Haus, der Eingang ist durch ein Gittertor gesichert, zu gelangen. Bei genauem Hinsehen sehen Sie einige Zentimeter hinter dem Tor einen Schlüssel liegen, doch leider sind Ihre Arme zu kurz, um durch das Tor hindurch den Schlüssel greifen zu können.

Wie wollen Sie in das Haus hineingelangen ?

Nun, überklettern des Tores ist nicht möglich, aber einige Schritte vor dem Haus waren Ihnen einige Bäume aufgefallen. Irgendwann im Laufe des Spieles haben Sie außerdem ein Kaugummi gefunden.

Ist der Groschen gefallen ?

Wir haben auch einige Zeit für die Lösung dieses Problemes gebraucht, vor allem, weil uns niemand so deutlich auf verwendbare Gegenstände hingewiesen hatte:

Man brauchte einfach nur zu dem Baum zu gehen und einen Ast abzubrechen. Dann ging man zum Eingang

des Hauses zurück, kaute das Kaugummi gut durch, befestigte es an einem Ende des Stockes, schob dieses Ende durch das Gitter des Tores und berührte den Schlüssel, worauf dieser am Kaugummi kleben blieb; durch vorsichtiges Zurückziehen des Stockes gelangte man schließlich und endlich in den Besitz des Schlüssels.
Welch ein Glück, daß dieser dann auch zu dem Tor paßte.

Eine elegante Lösung, nicht ganz einfach, aber vom Programmierer schon während der Entwicklungsarbeit genau so geplant. Ich sollte allerdings ergänzen, daß in der dem Spiel beigefügten Anleitung einige Spielerfahrung vorausgesetzt wurde.

Vielleicht hat obiges Beispiel Sie bereits in ´Abenteuerlaune´ versetzt, sollten Sie die Kraft des geschriebenen Wortes zur Faszination jedoch noch nicht bemerkt haben, dann führen Sie sich bei nächster Gelegenheit doch einmal die *Unendliche Geschichte* zu Gemüte.

Welche Steigerungsmöglichkeiten werden sich nun durch die Implementierung guter Romane in Computersysteme ergeben ?

Vielleicht wird es sich eines Tages als notwendig erweisen, die Erläuterung des Begriffes ´Abenteuer´ in unseren Lexika zu ergänzen:

> Im 20.Jahrhundert ermöglichte die fortgeschrittene Computertechnik den Abenteuerroman zum Mitspielen. Als Meilenstein gilt Scott Adams "Adventureland"; es folgten zahlreiche weitere sogenannte Adventures, die bis zu perfekten Weltsimulationen entwickelt wurden. Zahlreiche Buchautoren mit bekannten Namen (z. B. Michael Crighton) erkannten rechtzeitig die neuen Möglichkeiten und schufen mit ihren Werken für viele Menschen einen Ausgleich zur Alltagswelt.

GESCHICHTE UND ENTWICKLUNG DER ADVENTURES

Nachdem aus der ersten Rechenmaschine ein programmgesteuertes Rechengehirn geworden war, setzten in Programmiererkreisen die unterschiedlichsten Bemühungen ein, den Computern menschliches Denken und menschliche Verhaltensweisen beizubringen. Im Jahre 1966 gelang es schließlich Joseph Weizenbaum vom Massachusetts Institute of Technology mit seinem Programm ELIZA, nicht nur die Fachleute, sondern auch die Öffentlichkeit zu interessieren.
Eliza, in abgemagerten Versionen heute auch für jeden Microcomputer erhältlich, simuliert einen Psychiater und ahmt dessen typische Gesprächstechnik nach. Bei Versuchen mit Testpersonen zeigten sich diese nach der jeweiligen Sitzung sehr überrascht, einer Maschine ihre persönlichsten Probleme mitgeteilt zu haben.

Die weitere Entwicklung der Künstlichen Intelligenz brachte den Fachwissenschaftlern ein auf einer PDP-10 implementiertes Programm: ADVENTURE - Colossal Cave. Zur Abwechslung durften die gelehrten Herren auf einem anderen Gebiet forschen, nämlich nach Schätzen in einer düsteren, von Magie beherrschten Welt. Colossal Cave erfreute sich in Fachkreisen einer recht großen Beliebtheit und wurde schließlich im Sommer 1979 dann auch einer größeren Öffentlichkeit zugänglich gemacht. Die heute wohl jedermann bekannte Firma MICROSOFT veröffentlichte *Microsoft´s Adventure*, eine Diskettenversion dieses Uradventures für den damals in Amerika populärsten Microcomputer, den TRS-80.

Salonfähig gemacht wurden die Adventures jedoch schon im Jahre 1978 von einem jungen Amerikaner namens Scott Adams, welcher mit seinem ADVENTURELAND den Grundstock für den Newcomer ADVENTURE INTERNATIONAL legte.

Auch in diesem Spiel geht es darum, in einer mystischen Welt diverse Schätze zu finden, was natürlich durch Drachen, Labyrinthe, Lavaströme und sonstige Eigen- und Unarten erschwert wird. Dabei fängt alles ganz harmlos mitten im Walde an. Doch wenn man sich erst einmal orientiert hat und weiß, in welche Richtung man sich wenden muß, und dann auch noch den Eingang in das unterirdische Reich findet, dann ...

Die Nachfrage nach ADVENTURELAND war riesig, wohl auch deshalb, weil es noch keine Spiele in Spielhallenqualität gab, und so wurde wegen des großen Erfolges aus diesem ADVENTURELAND rasch eine Reihe von 12 Adventures, wobei durch geschickte Wahl der Thematik das Interesse der Kunden wachgehalten wurde.
Wollten Sie sich nicht auch schon immer einmal mit dem Grafen Dracula anlegen - in *The Count* können Sie es. Oder möchten Sie lieber einen Saboteur in einem Atomkraftwerk finden - kein Problem, *Mission Impossible* macht´s möglich.

Zu dieser Zeit erschien auf dem amerikanischem Softwaremarkt als Mitreiter auf der gleichen Welle ein Programmtyp, der als ´***INTERACTIVE FICTION***´ bezeichnet wurde. Diese Programme sollten einen neuen Literaturtyp darstellen, sollten, wie der Name sagt, Bücher zum Mitwirken sein. Zunächst ist der Spieler nur Leser, er wird nach Programmstart sehr genau in die Handlung eingeführt. Zahlreiche Seiten flimmern über den Bildschirm, die dem Leser Auskunft über Vorgeschichte, augenblickliche Situation, Hintergründe usw. geben, und die sich eben wie ein Buch lesen lassen.

So beginnt *Local Call For Death* wie eine Kriminalgeschichte: Drei Männer treffen sich in ihrem Clubhaus, im Verlauf des Abends wird einer von ihnen ans Telefon gerufen. Es ist der Neffe, welcher den Onkel, sichtlich erregt, um Geld bittet. Der Onkel versagt ihm jedoch die Unterstützung, weiß er doch aus Erfahrung, daß

das Geld auf der Rennbahn verwettet werden wird. Am nächsten Morgen erfährt er vom Selbstmord seines Neffen.

An dieser Stelle übernimmt nun der Spieler. Er stellt ab jetzt eine der Hauptpersonen dar, und versucht im direkten Gespräch mit den Mitwirkenden den vermeintlichen Selbstmord aufzuklären.

Typisch für diese Programme ist die klare Fixierung auf ein einziges Ziel, wie hier die Aufklärung eines Mordes.
Um dieses Ziel zu erreichen, müssen wir als Spieler uns weniger irgendwelcher Gegenstände als geschickter Fragen bedienen. Was ist wohl interessanter, in einem Buch ein Gespräch zu lesen, in einem Film ein Verhör zu verfolgen, oder das Erfolgserlebnis zu haben, durch eigenes Kombinationsvermögen einen Verbrecher überführt zu haben ?

Gerade diese Fähigkeit zu Dialogen macht den Reiz von Interactive Fiction aus, ein Musterbeispiel ist das ebenfalls bei Adventure International erschienene *Encounter in the Park*.
In diesem Spiel begegnen wir während des Morgenspazierganges unserer Traumfrau, und es wird unser erklärtes Ziel, eben dieses Mädchen zu verführen und dazu zu bringen, uns ihr Ja - Wort zu geben. Der Programmierer hat dem Spieler zur Erfüllung dieser Aufgabe ein breites Spektrum an Überredungskünsten offengelassen, und es macht wirklich großen Spaß, zu erleben, wie diese Dame auf gewisse Vorschläge reagiert.
Das schlimmste, was uns bei allzu unseriösen Vorschlägen passieren kann, ist, daß sie sich entrüstet von uns abwendet, und wir Mitteilung darüber erhalten, daß wir aus lauter Enttäuschung den Rest unseres Lebens im Kloster verbracht haben.

Wie bereits gesagt, war das Interesse an all diesen Textadventures sehr groß, immer mehr Firmen veröffentlichten demzufolge mehr oder weniger gute Kopien

dieser Spiele. Befleißigt durch den zunehmenden Druck der Konkurrenz, ließen sich daher einige Softwarehersteller neue Generationen von Adventurespielen einfallen.
Es erschienen die ersten ***LABYRINTH - ADVENTURES***, welche den Spieler ihrer Art entsprechend vor die Aufgabe stellten, lebend aus einem Gebäude zu entkommen. Fast immer endete ein Spiel damit, daß dieser sich hoffnungslos in dem perspektivisch auf dem Bildschirm dargestelltem Labyrinth verirrte.

Nächste Stufe der Entwicklung waren die ***QUEST - ADVENTURES***, die dann meistens auch gleich als Real - Time Adventures implementiert wurden. Bei diesem Sproß der großen Adventurefamilie wird der Spieler einem zusätzlichen Streßeffekt unterworfen, denn sollte er beim Erscheinen irgendwelcher Räuber oder anderer Unholde nicht sofort F für Fight eingeben und danach laufend irgendwelche Tasten oder gar Tastenkombinationen zu dem Zwecke zu drücken, sein Schwert in genau definierte Richtungen zu schwingen, endete das Spiel bereits sehr frühzeitig wegen zu starker Beschädigungen der Hauptperson.
Leider ergibt sich aus der Konzeption dieser Spiele eine starke Einschränkung des Handlungsspielraumes des Spielers, denn es bleiben diesem nur einige wenige Aktionen wie Kämpfen, Verhandeln, Kaufen oder Flüchten, die ihm per Menü, daher QUESTion, angeboten werden.

Die nächste Entwicklungsstufe der Adventurespiele wurde dank des Preisverfalls für Grafikprozessoren und RAM - Bausteine möglich, denn nun war es für die Computerhersteller nicht mehr schwierig, dem Heimanwender bei preisgünstigen Computern eine grafische Leistung zur Verfügung zu stellen, die früher nur auf vielfach größeren und somit auch teureren Anlagen möglich war.

So begann im Jahre 1982 (für Apple - Benutzer etwas früher) mit dem Vertrieb des ersten ***GRAFIKADVENTURES*** ein neuer Aufschwung für diese Spiele. Denn waren sie von vielen

Computerbesitzern als reine Textspiele bislang verpönt worden, so waren sie nun wegen der vielen bunten Bilder auf einmal interessant; darüber hinaus erwiesen sie sich als hervorragend dazu geeignet, der computerunkundigen Verwandschaft auf die Frage *'Was kann denn dein Computer ?'* zumindest ein erstauntes *'Oh, ist ja schön !'* zu entlocken.

An Spielwert und Ziel hatte sich natürlich überhaupt nichts geändert.
Plötzlich waren jedoch weitere Käuferschichten erreichbar; ein Grund, viele jahrelang bekannte Adventures im neuen Gewande auf den Markt zu werfen: die ausführlichen Beschreibungen waren verschwunden, statt dessen waren die augenblicklichen Gegebenheiten auf dem Fernsehschirm zu sehen.

Ob die Programme dadurch besser oder einfacher spielbar geworden waren sei dahingestellt, ich persönlich ziehe es jedoch vor, durch eine Mitteilung wie

Ich bin im Wald. Um mich herum stehen lauter Bäume. Zwischen zwei Eichen sehe ich ein Erdloch.

auf wichtige Dinge hingewiesen zu werden, als im Unterschied dazu nur einige Bäume zu sehen; diese halte ich dann verfrüht für ein ganz normales Stück Wald, weshalb ich auch schnell weitergehe, die Bäume nicht weiter untersuche und dadurch das Erdloch, einen kleinen schwarzen Flecken auf dem Monitor, natürlich übersehe. Selbstverständlich liegt ausgerechnet darin ein Teil des Schatzes oder aber ein Gegenstand, ohne den eine erfolgreiche Beendigung des Spieles nicht möglich ist.

2. KAPITEL
- DAS KONZEPT -

In den folgenden Kapiteln dieses Buches werden wir gemeinsam Adventurespiele entwickeln. Selbstverständlich wünschen wir für unser Werk ein möglichst professionelles Aussehen, daher werden wir nicht darauf verzichten können, Eigenschaften und spezielle Features auf dem Markt befindlicher Programme zu analysieren.
Anschließend können wir uns dann Gedanken darüber machen, wie wir die einzelnen Funktionen in BASIC programmieren.
Dabei wollen wir darauf achten, daß diese einzelnen Routinen dermaßen aufgebaut sind, daß sie nicht nur für ein bestimmtes Adventure brauchbar, sondern ohne Änderungen für den Einsatz in allen unseren Abenteuerprogrammen geeignet sind.

DIE AUFMACHUNG

Es wurde bereits herausgestellt, daß ein Adventure den Spieler in eine andere Welt versetzen will, in eine Welt, die das Unmögliche möglich macht. Er soll hier Abenteuer erleben, muß sich also bewegen und handeln können, muß über die Resultate seiner Bemühungen informiert werden, damit er weitere Handlungen vornehmen kann.
Sinnvolles und zielgerichtetes Handeln ist dem Menschen jedoch erst nach Gebrauch seiner Sinne möglich. Die Informationen, die der Abenteurer im wirklichen Leben mit seinen Augen, Ohren und seinem Tastsinn aufnimmt, müssen dem Spieler daher in geeigneter Form angeboten werden.
Da das menschliche Gehirn bis heute nicht direkt an einen Computer angeschlossen werden kann, müssen uns Antworten auf die Fragen *Wo bin ich ? Was sehe ich ? Wohin kann ich gehen ? Was fühle ich ?* und *Was sehe ich ?* auf andere Art und Weise zugänglich gemacht werden, was hier in Gestalt kurzer, aber vollständiger Sätze geschieht.

Die Erfassung der Umwelt wird allerdings in den seltensten Fällen bewußt ablaufen, Rauminhalte werden ohne besonderen

Wunsch innerhalb von Sekundenbruchteilen registriert, daher kann auch dem Adventurespieler nicht zugemutet werden, zunächst umfangreiche Abhandlungen über den gerade betretenen Raum zu studieren.
Selbst wenn es nur fünf oder zehn Sekunden wären, die er mit dem Lesen von mehreren Zeilen Beschreibungen zubrächte, würde die dazu erforderliche Konzentration den Spielfluß hemmen und es dem Spieler unmöglich machen, sich mit der Spielfigur zu identifizieren und sich der Illusion einer Scheinwelt hinzugeben.

Um dieses Problem zu umgehen, beschränkt man die Textausgabe auf ein akzeptables Minimum an Informationen und präsentiert diese, sofern es sich nicht um Grafikadventures handelt, in zwei übersichtlichen Zonen auf dem Monitor, von denen jede eine genau definierte Funktion hat.

So wird uns die obere Bildhälfte darüber Auskunft geben, wo wir uns gerade befinden und was wir sehen, die untere Hälfte ist hingegen für die Kommunikation mit der für uns handelnden Hauptperson des Adventures bestimmt. Hier geben wir unsere Anweisungen ein und hier erfahren wir, was auf unsere Eingaben hin geschieht.

Ein typisches Schirmbild könnte etwa so aussehen:

Ich bin in einem düsteren Wald.
Ich sehe viele alte Bäume.

Ich kann nach Norden, Osten, Westen.

Was soll ich tun ?

Für den Fall, daß wir beabsichtigen, in dem Spiel möglichst rasch weiterzukommen, wäre es falsch, ziellos in eine der drei Himmelsrichtungen weiterzumarschieren.
Das Nächstliegendste, was ein Adventurespieler in jeder Situation, die für ihn neu ist, tun sollte, ist, alles auf das genaueste zu untersuchen. So wären mögliche Antworten auf *Untersuche Baum* etwa: *Es handelt sich um alte Eichen.* oder *Auch hier greift das Baumsterben um sich.*
Diese Antworten helfen uns zwar nicht weiter, zeigen sie uns doch scheinbar nur, daß es doch nicht immer so wichtig ist, alles zu untersuchen, doch woher soll man die Antwort vorher wissen ? Denn auch eine Antwort wie *In einem Stamm sehe ich eine Öffnung.* wäre möglich gewesen. Vielleicht handelt es sich um einen alten Spechtbau, in welchem sich ein Teil des gesuchten Schatzes befindet; *Untersuche Öffnung* wird es an den Tag bringen.
Hätten wir keine Anhaltspunkte gefunden, auch auf *Untersuche Wald* hin - es könnte ja ein Zauberwald sein - nicht, müßten wir uns entscheiden, wohin wir uns nun sinnvollerweise begeben wollen.
Wie würden Sie sich in solch einer Situation verhalten, ohne Karte und Kompaß, hilflos und allein, weitab aller Wege ?
Vermutlich würden Sie auf einen Baum klettern, um zu sehen, was sich in der näheren und weiteren Umgebung befindet.

Ihre Eingabe sollte also lauten: *Besteige Baum*, und wenn Sie nicht eine Mitteilung in der Art *Dazu bin ich nicht sportlich genug !* erhalten, wird sich das Bild auf dem Monitor ändern:

Ich bin in der Krone einer alten Eiche.
Ich sehe im Norden ein Gebirge,
im Osten einen See,
im Westen steigt Rauch empor.

Ich kann nach unten.

BESTEIGE BAUM

Okay !

Was soll ich tun ?

Damit sieht die ganze Angelegenheit doch schon viel besser aus, denn nach Eingabe von *U* für Unten haben wir nur noch das Problem, zu entscheiden, welche Gegend wir zuerst besuchen wollen.

Die Mitteilung *Okay !* soll uns darüber informieren, daß die Eingabe verstanden und ausgeführt worden ist. Wäre dem nicht so, hätten wir beispielsweise *Ich verstehe das Verb nicht !* erhalten, womit uns das Programm sicherlich zu einer anderen Ausdrucksweise veranlassen würde. Gleiches gilt selbstverständlich für das Objekt unserer Eingabe.

Denkbar sind allerdings auch noch zwei weitere Antworten. Zum einen werden wir während eines Spieles oft die Mitteilung *I must be stupid, but ... - Ich verstehe nicht, was Du meinst !* erhalten, womit das Programm uns mitteilen will, daß es unsere Eingabe für recht sinnlos hält, was bei *Töte Baum* zweifellos der Fall wäre. Jedenfalls ist diese Aktion dann nicht im Spiel vorgesehen.
Ähnlich verhält es sich mit der zweiten Standardmeldung *You can´t do that ... yet. - Das geht im Moment nicht.* signalisiert uns, daß die benutzten Worte zwar zum programmierten Wortschatz gehören, der Befehl im Moment jedoch nicht sinnvoll ist: *Nimm Schlüssel* wird vom Adventureprogramm verstanden werden, wenn ein Schlüssel im Adventure auftritt, wird jedoch sinnlos, solange sich dieser Schlüssel nicht im gleichen Raum wie der Spieler befindet.

Dennoch dürfen wir aufatmen, denn sollten wir diese Antwort erhalten, sind wir auf dem richtigen Weg; es sind nur noch nicht alle erforderlichen Bedingungen erfüllt, um den Befehl auszuführen. So wird sich eine verschlossene Tür öffnen lassen, wenn wir uns ein für diesen Vorgang geeignetes Instrument beschafft haben. War die Antwort auf *Öffne Tür* allerdings *Ich verstehe nicht, was Du meinst.* dann wären alle weiteren Bemühungen zwecklos.

VORÜBERLEGUNGEN

Der generelle Aufbau der Adventurespiele ist uns nun bekannt. Wir haben eine Vorstellung davon, wie unser Adventure sich auf dem Monitor zu präsentieren hat und welche Informationen bereitgestellt werden müssen. Im folgenden wollen wir uns näher mit den einzelnen Elementen unseres Adventuresystems beschäftigen und eine Konzeption ausarbeiten, die im nächsten Kapitel realisiert wird.

Räume im Adventure:

Die Adventurewelt setzt sich aus den Räumen des jeweiligen Spieles zusammen, d.h. jeder durch den im Adventure herumwandernden Spieler erreichbare Ort wird als Raum bezeichnet. Dabei ist dessen Größe völlig unbedeutend, es kann, wie in unseren bisherigen Beispielen, ein Wald, bzw. eine Baumkrone gemeint sein, es kann sich jedoch auch um das Innere eines Autos oder Kleiderschranks handeln.
Die einzelnen Räume unterscheiden sich für den Spieler durch ihre Beschreibungen und ihre geografische Anordnung; programmtechnisch unterscheiden sie sich, wie wir noch sehen werden, durch ihre Raumnummer.

Alle diese Räume sind für den Spieler erreichbar, müssen also in geeigneter Weise miteinander verbunden werden. In jedem Adventure ist die Bewegung in die vier Himmelsrichtungen Norden, Süden, Westen und Osten möglich, einige erlauben zusätzlich Bewegungen in der Vertikalen.
Selbstverständlich darf es jedoch nicht geschehen, daß der Spieler Raum 5 in westlicher Richtung verläßt, Raum 6 dabei von Osten her betritt, und wenn er noch einmal in Raum 5 will, keinen Weg in Richtung Osten findet, oder, schlimmer noch, wenn er durch eine Bewegung nach unten wieder in Raum 5 gelangt.
Um der Kritik von erfahrenen Abenteurern vorzubeugen, Ausnahmen bestätigen auch hier die Regel, denn was wäre ein Adventure ohne ein magisches Labyrinth.

Objekte

Typisch für jeden Raum sind weiterhin die in ihm befindlichen Objekte. Wir müssen alle Gegenstände in den Räumen unterbringen, oder, korrekter ausgedrückt, jedem Objekt einen Raum zuordnen. Bei dieser Zuordnung vermeiden wir automatisch, daß irgendwelche Objekte doppelt existieren.

Ein weiteres Problem stellt sich uns mit der Unterbringung derjenigen Objekte, die bei Spielstart gar nicht im Adventure vorhanden sein dürfen. - Wo bleiben sie bis zu ihrem Auftreten ?

Angenommen, unser Held betritt einen Raum, und als sichtbares Objekt wird nur eine alte, verschlossene Holzkiste ausgegeben.
Auf die Eingabe des Spielers *Öffne Kiste* verlangen sämtliche Regeln der Logik, daß die alte, verschlossene Kiste vom Bildschirm verschwindet, dafür sollte jedoch eine

alte, geöffnete Kiste auftauchen, die womöglich mit Silbermünzen gefüllt ist.

Nun, die Lösung dieser Probleme wird uns keine allzu großen Schwierigkeiten machen. Wir werden in unserem Adventure einen zusätzlichen Raum, den *Lagerraum*, installieren und in diesem alle zur Zeit nicht benutzten Objekte lagern. Wenn wir keine Verbindungen zu diesem Raum hin programmieren, kann der Spieler ihn auch nie betreten und somit auf unerwünschte Weise mit den betreffenden Gegenständen in Kontakt kommen.

Im nächsten Kapitel werden wir sehen, wie einfach der Lagerraum und auch die gesamte Adventurewelt zu programmieren ist; weiterhin werden wir im Verlauf des Buches die Notwendigkeit weiterer besonderer Räume einsehen.

Eingaben

Wie bereits aus dem voranstehenden Text hervorgegangen ist, bestehen die Eingaben des Spielers meist aus einem Verb und einem Objekt. Unser Programm wird daher zunächst testen müssen, ob eine Eingabe erlaubt ist, wozu eine Trennung in Verb und Objekt erforderlich sein wird. Sollte der Spieler sich verschrieben haben, oder das benutzte Verb nicht vorgesehen sein, muß eine entsprechende Meldung gemacht werden. Findet unser Adventure das Eingabeverb innerhalb seines Wortschatzes wieder, muß das Objekt auf gleiche Weise überprüft werden.
Sind beide Worte definiert, kann das Programm in vorgeschriebener Weise reagieren.

Bei der Erstellung dieses Wortschatzes muß der Programmierer übrigens besondere Sorgfalt walten lassen, er

muß alle möglichen Eingaben, die irgendein Spieler machen könnte, vorhersehen.
Wie schwierig gerade dieses Unterfangen werden kann, bewies mir kürzlich einer meiner Bekannten, der nie zuvor ein Adventure gespielt hatte, und der, nachdem er das Titelbild zusammen mit den notwendigsten Instruktionen studiert hatte, alle Probleme auf einen Schlag mit *Finde Schatz* lösen wollte, einer Eingabe, mit der ich als Programmierer natürlich nicht gerechnet hatte.

Nebenbei wird auch die Einbringung von Synonymen erforderlich werden, denn sollte der Spieler allzu lange damit beschäftigt sein, dem Adventure seine Wünsche klarzumachen, wird er schnell das Interesse verlieren.

Gerade für uns Deutsche kann die Wahl der Worte jedoch zum Problem werden, denn wo der Amerikaner sich mit *Climb Tree, Shoot Gun, Drop Box, Look* leicht verständlich machen kann, sieht es für uns nicht ganz so eindeutig aus: *Kletter Baum, Schieß Gewehr, Leg Kiste, Sieh.*

Mitteilungen

Die erste Ausgabe auf dem Bildschirm, die nicht mehr zu der Beschreibung des Raumes gerechnet werden kann, informiert den Spieler über die Himmelsrichtungen, in deren Richtung er den Ort des Geschehens wieder verlassen kann.

Weiterhin müssen die Ausführung einer jeweiligen Aktion, wie auch eine eventuell darauf hin stattfindende Reaktion, dem Spieler deutlich gemacht werden. Im einfachsten Fall geschieht dies durch O.K., meist werden jedoch zusätzliche Mitteilungen ausgegeben, wobei die unterschiedlichsten Eingaben oft die gleiche Antwort erforden werden, Antworten, die aus eben diesem Grunde in einer Liste geführt, und standardisiert werden sollten. Beispielsweise

werden *So stark bin ich nicht.* oder *Das hat doch wohl keinen Sinn!* und *Ich verstehe nicht, was Du meinst!* recht häufig auftauchen.

Standardfunktionen der Adventures

In einem durchschnittlichen Adventure finden wir 30 - 50 verschiedene Räumlichkeiten, in jedem Raum einen oder mehrere Gegenstände. Diese Gegenstände können wiederum irgendwelche Sachen enthalten, und da man im Adventure nie genau weiß, was ein paar Züge weiter unbedingt gebraucht wird, neigen viele Adventurespieler dazu, alles mitzunehmen, was ihnen zwischen die Finger kommt und nicht zu schwer zum Tragen ist.
Den Überblick zu behalten, wird dabei recht schwierig, denn da wir alle bequem geworden sind, machen wir uns natürlich keine Notizen.
Diese Erkenntnis kam den Programmierern glücklicherweise recht früh, so daß heute jedes Adventure nach Eingabe von *INVENTUR* alle Gegenstände, die wir mit uns führen, auf dem Bildschirm ausdruckt.

Falls es sich um ein freundliches oder speziell für Anfänger geschriebenes Adventure handelt, können wir uns in verfahrenen Situationen Tips geben lassen, denn dann ist der Befehl *HELP - HILFE* implementiert.

Im Gegensatz zur Standardeingabe aus Verb und Objekt handelt es sich hier, wie übrigens auch bei Inventur, um einen sogenannten Ein - Wort Befehl, welcher in diesem Fall jedoch nicht einmal eine Aktion einleitet.
Doch was kann dieses eine Wort nicht alles bewirken:

Ich würde alles untersuchen zeigt uns unsere Nachlässigkeit, die Mitteilung *Eine feenhafte Gestalt*

erscheint und schreibt den Satz 'Sesam öffne dich' in den Sand wird uns hingegen in einer verfahrenen Situation zu einem Freudensprung veranlassen.
Schlechter sieht es aus, wenn wir für die Hilfe bezahlen sollen: *Ein Troll erscheint und sagt, daß uns für seine Hilfe 10 Punkte abgezogen werden. Er will wissen, ob wir Hilfe benötigen ?*
Dunkle Wolken werden vermutlich in den Fällen am Horizont auftauchen, wenn wir auf die Eingabe von Hilfe folgende Nachricht erhalten: *'Hilfe' ist in diesem Adventure nicht vorgesehen !* oder wenn als Antwort nur die allgemeine Spielanweisung wiederholt wird.

Sicherlich mag es ärgerlich sein, wenn man an einer gewissen Stelle trotz aller Mühen nicht weiterkommt, jedoch sollte niemand die letzten zwei Beispiele als einen Akt der Boshaftigkeit des Programmierers ansehen; wäre die Hilfsfunktion durchgehend gewährleistet, würden einige ganz Schlaue das Adventure in 30 Minuten lösen, dabei aber niemals die Faszination eines solchen Programmes kennenlernen.
Einige Programme achten daher auf den Gebrauch dieser Funktion und verweigern bei allzu häufiger Anwendung schließlich jede Hilfe oder lenken den Spieler gar auf falsche Fährten.

3. KAPITEL
- DIE VERWIRKLICHUNG -

HINWEISE ZU DEN LISTINGS

Bevor wir mit der Programmierung beginnen, möchte ich Ihnen noch einige Hinweise eher technischer Natur geben, die mit dazu beitragen sollen, daß Ihnen der Spaß an der Arbeit mit diesem Buch nicht durch irgendwelche auftretenden Fehler vergällt wird.

Sicherlich werden die folgenden Programmzeilen nicht die ersten sein, die Sie mühsam über die Tastatur eingeben, und so werden Sie auch die Erfahrung gemacht haben, daß sich trotz aller Sorgfalt immer wieder Tippfehler einschleichen.

Geradezu prädestiniert für Fehler dieser Art sind die DATA - Zeilen, besonders, wenn diese nur Ziffern enthalten wie es auch hier der Fall sein wird.
Weiterhin ist es in einigen Zeilen unerläßlich, diese *genau* so einzugeben, wie sie abgedruckt sind. Dies gilt insbesondere für die Anzahl der Blanks (Leerzeichen) in den Strings (Zeichenketten).

Um Ihnen eine wirksame *Kontrollmöglichkeit* zu geben, sind alle Listings in diesem Buch so abgedruckt, wie Sie sie *nach Eingabe von LIST* auf Ihrem Bildschirm sehen müßten.

Überprüfen Sie daher, wenn Sie den Eindruck gewinnen, daß irgendetwas wohl nicht wie vorgesehen funktioniert, zunächst nur jeweils die ganz rechts in einer Zeile stehenden Buchstaben, stimmen sie nicht überein, haben Sie mit allergrößter Wahrscheinlichkeit bereits die fehlerhafte Programmzeile gefunden.

Darüber hinaus sollten Sie nur die vorgegebenen Zeilennummern verwenden, was den Erhalt der Funktionsfähigkeit Ihres Programmes garantiert.
Denn mit der weiteren Entwicklung werden einige Programmzeilen ausgetauscht wie auch zusätzliche Routinen eingefügt werden.

DIE SPIELIDEE

Damit haben wir das Mindestmaß an grauer Theorie bewältigt, alles was uns noch fehlt, ist eine Geschichte, die wir als Abenteuerspiel realisieren wollen. Dann können wir unseren CPC 464 einschalten und mit der Praxis beginnen. Vielleicht haben Sie bereits eine feste Vorstellung von Ihrem ersten Adventure, wenn nicht, lassen Sie sich doch durch folgende Vorschläge inspirieren.

1. *DAS SPUKHAUS*

Sie erhalten den letzten Brief von Ihrer Tante, ein Schreiben, in dem sie Ihnen ihr altes hochherrschaftliches Haus vererbt. Stutzig macht Sie jedoch der Hinweis, daß Sie sich vorsichtig verhalten sollen, denn sonst könne es Ihnen wie Ihrem Onkel ergehen. Sie lassen sich durch solcherlei Gerede nicht davon abhalten, das Haus aufzusuchen. Sogleich stellen Sie allerlei Ungereimtheiten fest, finden eine geheime Bibliothek mit Büchern über Geisterbeschwörung, entdecken im Keller eine Opferstätte. Zahllose Geister machen Ihnen das Leben schwer, bis Sie schließlich entdecken, daß einer Ihrer Vorfahren den Sektenführer zum Ausbau der eigenen Macht ermordet hatte und zur Strafe lebendig eingemauert worden war. Sie sollen ihm nun zur ewigen Ruhe verhelfen.

2. *DAS VERMÄCHTNIS DES ALTEN*

Ihr Freund und Nachbar, ein berühmter Wissenschaftler, ist unter mysteriösen Umständen ums Leben gekommen. Sie erhalten eine Notiz, in der er Sie bittet, sein Lebenswerk zu vollenden. Nachdem Sie es geschafft haben, sein bislang geheimes Labor zu betreten, ist es Ihre Aufgabe,

seinen Supercomputer in Betrieb zu nehmen. Dieser gibt Ihnen dann weitere Hinweise.

3. *STAR ODYSSEY*

Ihr Raumschiff wird durch einen kosmischen Sturm so stark beschädigt, daß Sie auf einer unbemannten Station landen müssen, um Ersatzteile zu beschaffen, damit Sie zur Erde zurückkehren können.

4. *GOLDRAUSCH*

Sie treffen in der Wildnis auf einen tödlich verletzten alten Mann. Dieser macht Ihnen Mitteilung über seine Goldmine. Er zeigt Ihnen einige Goldstücke und berichtet Ihnen, daß er an sieben verschiedenen Stellen auf dem Minengelände weiteres Gold versteckt hätte; da er es nicht mehr benötige, könnten Sie es sich holen.

Diesen zuletzt gemachten Vorschlag, ein Adventure des Typs Schatzsuche, möchte ich als Grundlage für die nächsten Kapitel dieses Buches nehmen, und gemeinsam mit Ihnen ausarbeiten. Selbstverständlich steht es Ihnen frei, eine eigene Idee zu realisieren, jedoch werde ich bei allen noch vorzustellenden Erweiterungen immer wieder Bezug auf *Goldrausch* nehmen.

DIE GESTALTUNG DER WELT

Mit der Wahl des Themas steht der grobe Ablauf des Adventures bereits fest. Im folgenden werden wir einzelne Strukturen ausarbeiten und diese immer weiter verfeinern, bis eine bis ins kleinste geplante Welt geschaffen worden ist.

Genauso wollen wir auch bei der Programmierung verfahren; Schritt für Schritt werden wir unser Adventure aufbauen und uns immer wieder von der korrekten Funktion der einzelnen Routinen überzeugen.

DIE KARTE

Am Anfang steht zunächst das Nichts. Doch werden wir die für unser Spiel so dringend benötigte Welt im folgenden nach unseren Vorstellungen schaffen. Was unsere Welt bieten muß, legt das Thema fest. Goldrausch wird vermutlich in und um einem Bergwerk stattfinden.

Stellen wir uns die Mine einmal vor: ein zerklüfteter Abhang voller Geröll; ein Werkzeugschuppen; Holzrinnen, die das für die Goldwaschung benötigte Wasser eines Gebirgsbaches heranführen; morsche, trockene Holzbalken, die den düsteren Eingang säumen; dunkle Gänge voller Gefahren.

Sicherlich werden uns ohne große Schwierigkeiten noch zahlreiche ähnliche Bilder einfallen, die uns eine für die vorgesehene Handlung ausreichende Menge Räume finden lassen werden.

Unser konkretes Beispiel möchte ich zunächst auf nur sechs Räume beschränken. Beginnen wollen wir mit einer Wanderung

durch einen Wald bis wir zur Mine gelangen, wo unsere Aufgabe darin bestehen wird, die Schätze zu suchen.

Damit ist der vorläufige Beginn von Goldrausch festgelegt. Um jedoch über Sieg oder Niederlage des Spielers entscheiden zu können, müssen wir ebenfalls ein erfolgreiches Spielende festlegen, mit weiterem Fortschreiten unseres Projektes natürlich auch mehrere verlustbringende Situationen.

So können wir uns mit dem Finden aller Schätze begnügen, können dem Abenteurer aber auch die Aufgabe stellen, den gefundenen Schatz an einem sicheren Ort zu deponieren. Zunächst soll das Auffinden des Goldes jedoch ausreichend sein.

Aufgabe und Ziel stehen fest, ebenso zwei wesentliche Handlungsorte. Da ist zum einen der Wald, in dem wir starten und den wir aus zwei Räumen aufbauen werden, und zum anderen die Mine. Allerdings ist uns klar, daß wir in dieser kleinen Welt keine Handlung ablaufen lassen können. Denn wir brauchen Platz, um einige Gegenstände verstecken und dem Spieler einige Fallen stellen zu können.

Normalerweise wird ein Bergwerk wohl auch kaum zwischen den Wurzeln eines Baumes beginnen, wir müssen geeignete Übergänge der Landschaft schaffen, um das Spiel realistisch zu gestalten.

Drei weitere Räume stellen uns zur Ausarbeitung der ersten Version unseres Adventures genügend Platz zur Verfügung, wir können nun eine Liste aller auftretenden Orte formulieren:

im Wald
im Wald
durch einen Felshang begrenzter Waldrand

eine Waldlichtung
Waldlichtung am Berghang
Eingang ins Bergwerk

Der nächste Schritt zum fertigen Programm ist zweckmäßigerweise die Anfertigung einer Karte, in der alle Orte als Rechtecke dargestellt werden. Diese Rechtecke werden durchnumeriert und mit ihrer Raumbeschreibung versehen, wobei wir bereits auf eine passende Formulierung achten und auch nach Möglichkeit die Zeilenlänge von 40 Zeichen berücksichtigen, die wir auf unserem Schneider verwenden wollen. So muß sich unsere Beschreibung reibungslos an den einleitenden Text *Ich bin* anfügen.

Welche Nummer der einzelne Raum erhält, spielt keine Rolle. Wir müssen nur mit eins beginnen und fortlaufend weiterzählen, wodurch wir auch doppelte Numerierungen verhindern. Anschließend verbinden wir die Räume untereinander und notieren an jeder Begrenzung die entsprechende Himmelsrichtung.

Aus dieser Karte läßt sich nun ohne weiteres ablesen, daß, wenn der Spieler sich beispielsweise in Raum 3 befindet, folgende Mitteilungen auf dem Monitor ausgegeben werden müssen:

Ich bin im Wald vor einem Felshang.

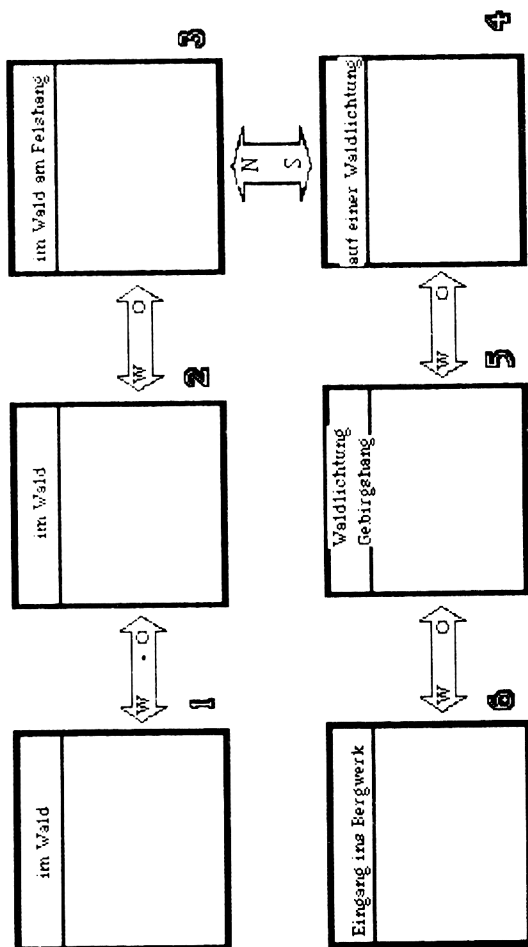
Weiterhin gibt uns unsere Karte Aufschluß über die möglichen Bewegungsrichtungen:

Ich kann nach Westen, Süden.

In unserem Programm könnten wir nun für jeden Raum eine Bedingung der Art *Wenn Spieler in Raum 3, dann drucke ´Ich bin im Wald vor einem Felshang.´* programmieren, würden

damit unseren Speicher aber schon nach kurzer Zeit gefüllt haben.

Das Prinzip ist jedoch richtig. Wir werden alle Räume in einem Variablenfeld speichern, und zur Ausgabe der Beschreibung eines Raumes auf das jeweils benötigte Element zugreifen.



EXKURS: VARIABLEN & FELDER

Um Zwischenergebnisse oder Daten eines Programmes zu speichern, werden in jeder Computersprache Variablen verwendet.
Diese Variablen verhalten sich wie kleine Fächer, in denen etwas zur Aufbewahrung abgelegt werden kann. Diese Fächer haben Namen, Kofferraum oder Schreibtischschublade wären möglich. Dabei gibt es dann Fächer gleicher Art: am Schreibtisch untereinander Schublade 1, Schublade 2, 3, usw. Jedes Fach heißt Schublade und unterscheidet sich vom nächsten nur durch seine Nummer. So auch in Basic: Wir können jeder Variablen einen eigenen Namen geben, oder auch gleichartige mit einem einzigen Namen und einer Indexnummer versehen, diese wird dann in Klammern hinter die Bezeichnung der Variablen geschrieben.
Dabei lassen sich zwei grundsätzliche Variablentypen unterscheiden. Die einen nehmen nur Zahlen auf, Zahlen, mit denen auch gerechnet werden kann; die anderen Variablen dienen der Aufnahme von Texten. Zusammensetzung und Länge der Texte spielen keine Rolle, genauer: wir wollen jedes Zeichen auf unserer Tastatur und Längen bis zu 255 Zeichen hintereinander erlauben.
Korrekte Beispiele wären: A, A1, A2, TEXT1 für numerische Varaiablen (nur Zahlen), und A$, A1$, A2$, TEXT1$ für Stringvariablen. Im Gegensatz zu diesen einfachen Variablen wären A(1), A(2), A(3) bzw. A$(1), A$(2), A$(3) dann die entspechenden indizierten Variablen.
Darüber hinaus erlaubt Ihnen der Schneider CPC 464 dann noch eine Differenzierung der numerischen Variablen in ganze Zahlen und Gleitkommazahlen.

Vielleicht sind Ihnen auch einige unserer Mitmenschen bekannt, die versuchen, ihre Wohnung durch das Aufstellen von mit kleinen Miniaturen gefüllter Setzkästen - sie wurden für die Aufbewahrung der kleinen Lettern in einer Druckerei geschaffen - zu verschönern; wir Anfänger stellen

uns jetzt einmal jedes einzelne Kästchen als eine Variable vor.
Um gleich ein praktisches Beispiel zur Hand zu haben, führen wir, allerdings besser nur in Gedanken, folgendes durch:
Aus unserer Adventurekarte schneiden wir die Rechtecke der einzelnen Räumè aus. Diese legen wir in jeweils einen Kasten, achten dabei jedoch darauf, daß die frühere Anordnung der Räume erhalten bleibt, d.h. ganz oben links liegt unser Wald, in der zweiten Reihe links außen liegt der Eingangsstollen.
Auf die Frage, wo das Bergwerk sei, könnten wir mit ´in der ersten Spalte der zweiten Reihe´ antworten, ebenso sind die Positionen aller anderen Räume durch die Koordinaten Reihe und Spalte festgelegt. Diese Anordnung wird in Computersprachen als zweidimensionales Feld bezeichnet, oder, viel schöner, als Array.
In Basic würden wir unseren Setzkasten als Variablenfeld KARTE$ ($ weil Texte gespeichert werden sollen) bezeichnen, und den einzelnen Variablen folgenden Inhalt zuordnen:
KARTE$(1,1) = "im Wald" erste Reihe u. erste Spalte
KARTE$(1,3) = "am Felshang" erste Reihe, dritte Spalte
KARTE$(2,1) = "Eingang Bergwerk" usw.

Ab Zeile 500 geben wir daher folgenden Programmtext ein:

```
500 raum$(1)="im Wald."
510 raum$(2)="im Wald."
520 raum$(3)="im Wald vor einem Felshang
."
530 raum$(4)="auf einer Waldlichtung."
540 raum$(5)="auf einer Waldlichtung."
550 raum$(6)="vor einem Bergwerksstollen
."
```

Damit existieren die Räume in unserem Computer, es fehlt uns noch die Spielfigur, welche sich in der programmierten

Welt bewegt und uns die Situation, in der sie sich gerade befindet, beschreibt. Ihr Aufenthaltsort wird natürlich von unseren Eingaben abhängig sein; wir führen daher eine weitere Variable, die wir SPIELER nennen wollen, ein, diese speichert die jeweilige Position des Spielers.

Sinnvolle Werte für diese Variable sind bislang die Zahlen eins bis sechs, denn PRINT RAUM$(SPIELER) druckt dann die jeweilige Beschreibung des Aufenthaltsraumes aus. Geben wir zu Testzwecken die nächsten Zeilen ein:

Achtung ! Ich möchte Sie noch einmal bitten, die angegebenen Zeilennummern unbedingt beizubehalten, da einige der Zeilen mit der weiteren Entwicklung verändert, wie auch nicht benutzte Nummern durch zusätzliche Programmzeilen belegt werden!

```
1140 PRINT "Ich bin ";
1150 PRINT raum$(spieler)
1390 INPUT"Spieler in welchen Raum";spie
ler
5000 GOTO 1140
```

Erkundungen unserer Welt werden dadurch möglich, doch ist die Art der Weiterbewegung keineswegs befriedigend. Wir wollen nicht wahllos von einem Raum in einen anderen springen, sondern streben eine gezielte Orientierung nach den Himmelsrichtungen an.
Normalerweise wird der gerade vom Spieler besuchte Raum mindestens einen, maximal sechs Ausgänge haben. An Raum 1 grenzt im Osten Raum 2 an, in die Richtungen Norden, Süden und Westen, ebenso nach oben und unten, führt kein Weg. Wenn wir darin übereinkommen, daß die Himmelsrichtungen

stets in der Reihenfolge N, S, W, O, Oben und Unten genannt werden, kann Raum 1 folgendermaßen spezifiziert werden:

1 im Wald -,-,-,X,-,-

Zweckmäßigerweise werden Ausgänge dann nicht durch ein X markiert, sondern es wird explizit der in dieser Richtung zu erreichende Raum aufgeführt:

1 im Wald 0,0,0,2,0,0
2 im Wald 0,0,1,3,0,0

Diese sechs Werte werden für jeden Raum in einer Variablen, nennen wir sie DURCHGANG, gespeichert. Da hier zwei verschiedene Werte (Raum und Richtung) den zu betretenden Raum kennzeichnen, ist die Speicherung in einem zweidimensionalem Feld angebracht, wobei die Reihe jeweils gleich dem Raum ist und die Spalten eins bis sechs den möglichen Wegen entsprechen.

RAUM	*N*	*S*	*W*	*O*	*OB*	*U*
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	0	4	2	0	0	0
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Es dürfte niemandem schwerfallen, diese Richtungstabelle (auch Travel - Table genannt) zu programmieren.

Anfänger, welche die nächsten Zeilen nicht verstehen, sehen sich bitte unser vorangegangenes Setzkastenbeispiel noch einmal an.

```
501 DURCHGANG(1,1)=0 : DURCHGANG(1,2)=0
502 DURCHGANG(1,3)=0 : DURCHGANG(1,4)=2
503 DURCHGANG(1,5)=0 : DURCHGANG(1,6)=0
```

Obige Zeilen programmieren den Durchgang von Raum 1 nach Raum 2 in östlicher Richtung. Entsprechend könnten wir mit den anderen Räumen verfahren, würden dabei jedoch sehr verschwenderisch mit unserem Speicherplatz umgehen (wie oft schreiben wir ´Durchgang´ ?), außerdem bereitet die immense Tipparbeit nicht allzu viel Vergnügen.

Aus diesem Grunde verzichten wir auf die Zeilen 501 bis 503, und rufen uns an dieser Stelle lieber die Befehle DATA und READ in die Erinnerung zurück.

EXKURS: READ DATA

Üblicherweise benutzen wir die INPUT - Anweisung, um Daten per Tastatur an ein laufendes Programm zu übergeben. Sie hat die Form INPUT Var, wobei Var eine beliebige Variable sein kann. Zur sequentiellen Eingabe mehrerer Daten kann eine ausreichende Anzahl Variablen, durch Kommata getrennt, angehängt werden: INPUT LÄNGE, BREITE, HÖHE kann somit drei Zahlen nacheinander zur weiteren Verarbeitung ins Programm übernehmen.
Sind die Daten bereits bei Programmstart bekannt, können sie direkt ins Programm eingebaut werden, was natürlich nur sinnvoll ist, wenn diese Daten für jeden Programmlauf stets gleich sind. Die dazu notwendigen Schlüsselworte lauten READ und DATA. INPUT wird durch READ ersetzt: READ LÄNGE, BREITE, HÖHE leistet daher die gleiche Funktion, die

Eingabedaten werden mit DATA 1,2,3 in einer beliebigen Programmzeile bereitgestellt.
Wichtig zu wissen ist, daß die Daten in der auftretenden Reihenfolge eingelesen werden, die Zahl der Daten in DATA - Zeilen muß daher gleich der Zahl der Variablen in READ - Anweisungen sein, andernfalls tritt eine Fehlermeldung auf bzw. werden nicht alle Daten berücksichtigt.
Sollen die Daten wieder mit dem ersten Element beginnend eingelesen werden, so ist der Befehl RESTORE zu verwenden.

Wir geben daher folgende Zeilen in unseren 464´er ein:

```
500 REM ************* RAUMBESCHREIBUNGEN
501 DATA"im Wald"
502 DATA"im Wald"
503 DATA"im Wald vor einem Felshang"
504 DATA"auf einer Waldlichtung"
505 DATA"auf einer Waldlichtung"
506 DATA"vor einem Bergwerksstollen"
```

Nun müssen die Daten an die Arbeitsvariablen überwiesen werden. Diese Zuweisung erfolgt durch READ raum$(raum), wobei raum natürlich nacheinander die Raumnummern eins bis sechs annehmen muß. Um Fehler zu vermeiden, muß die Anzahl der Räume bekannt sein; wir führen die Variable AR = Anzahl Räume ein. Zweckmäßigerweise sollte sie gleich zu Beginn des Programmtextes initialisiert werden, damit spätere Erweiterungen des Adventures leicht durchgeführt werden können.

```
110 ar=6
```

Innerhalb einer Schleife werden die Raumbeschreibungen eingelesen:

```
845 FOR raum=1 TO AR
850 READ raum$(raum)
870 NEXT raum
```

Ebenso wird unsere Richtungstabelle implementiert; um die Übersicht zu behalten, schreiben wir die Richtungswerte in die jeweiligen Datazeilen direkt hinter die Raumbeschreibungen:

```
500 REM ************* RAUMBESCHREIBUNGEN
501 DATA"im Wald",0,0,0,2,0,0
502 DATA"im Wald",0,0,1,3,0,0
503 DATA"im Wald vor einem Felshang",0,4
,2,0,0,0
504 DATA"auf einer Waldlichtung",3,0,5,0
,0,0
505 DATA"auf einer Waldlichtung",0,0,6,4
,0,0
506 DATA"vor einem Bergwerksstollen",0,0
,0,5,0,0
```

Die möglichen Auswege müssen daher gemeinsam mit den Beschreibungen eingelesen werden. Da auf jeden Raum sechs Richtungsdaten folgen, wird eine zweite Schleife innerhalb der ersten aufgebaut:

```
855 FOR richtung=1 TO 6
860 READ durchgang(raum,richtung)
865 NEXT richtung
```

Somit ist jede Reihe unseres Arrays mit einem Raum identisch; in den sechs Spalten einer jeden Reihe ist der

Raum gespeichert, den der Spieler durch Eingabe der betreffenden Richtung betreten kann.

Durch diese Anordnung wird uns die im folgenden zu programmierende Fortbewegung des Spielers auf sehr einfache Art und Weise möglich sein.
Bevor dieser sich aber überhaupt irgendwohin wenden kann, muß er sich über die ihm zur Verfügung stehenden Auswege informieren können. Wir müssen also einige Programmzeilen entwickeln, die die freien Himmelrichtungen im Klartext auf dem Bildschirm ausdrucken.

Nehmen wir einmal an, unsere Hauptperson befindet sich augenblicklich in Raum 3.
Dann wird die Reihe 3 unserer Richtungstabelle angesprochen:

RAUM	***N***	***S***	***W***	***O***	***OB***	***U***
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	***0***	***4***	***2***	***0***	***0***	***0***
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Alles was wir tun müssen, ist, die Bezeichnungen derjenigen Spalten auszugeben, die keine 0 enthalten; in unserem speziellen Beispiel die Spalten zwei und drei, Süden und Westen.

Wie wir festgestellt haben, ist die Reihe des Feldes mit dem zu untersuchenden Raum identisch; zur Bereithaltung dieser aktuellen Raumnummer hatten wir zuvor die Variable SPIELER eingeführt. Zur Ausgabe der nicht versperrten Richtungen muß unser Programm somit die sechs Richtungen

des Raumes SP testen, und die Namen aller Spalten ausdrucken, welche einen Wert ungleich Null haben:

Sechs Richtungen müssen überprüft werden:

```
1250 FOR richtung=1 TO 6
1260 IF durchgang(spieler,richtung)<>0 T
HEN PRINT"** AUSGANG **"
1310 NEXT richtung
```

Bei einem Probelauf wird unser Programm jetzt jeden Raum beschreiben sowie jeden möglichen Ausgang anzeigen; leider aber noch nicht dessen Richtung. Deshalb bereiten wir ein weiteres Feld vor, ein Feld mit den Bezeichnungen der Richtungen (richtung$):

```
1020 DATA Norden, Sueden, Westen, Osten,
 Oben, Unten
1030 FOR richtung=1 TO 6
1040 READ richtung$(richtung)
1050 NEXT richtung
```

In den einzelnen Elementen sind die Bezeichnungen der sechs Richtungen gespeichert, und wir können uns das Feld RICHTUNG$ als eine Art Schablone vorstellen, die während des Programmlaufs auf die gerade aktuelle Reihe unserer Richtungstabelle gelegt wird:

```
1240 PRINT"Ich kann nach ";
1260 IF durchgang(spieler,richtung)<>0 T
HEN PRINT richtung$(richtung)
```

Zur Kontrolle starten wir unser Programm und geben einige Raumnummern ein - ein Blick auf unsere Karte wird uns davon überzeugen, daß wir bisher alles richtig gemacht haben.

Nach diesen Vorbereitungen werden wir nun die gezielte Bewegung in unserem Adventure möglich machen.
Jeder, der bereits einmal ein Abenteuerspiel geladen hatte, wird wissen, daß die Eingabe einer Himmelsrichtung zum Zwecke der Fortbewegung die wohl häufigste Anweisung an die Spielfigur ist. Daher sollten wir als Programmierer dem Spieler soweit entgegenkommen, daß wir die Eingabe des Anfangsbuchstabens als ausreichend betrachten, womit wir dem Abenteurer das Ausschreiben der Himmelrichtungen, und somit mehrere hundert Tastendrücke, ersparen.
Tatsächlich lassen sich zahlreiche Abenteuerspiele finden, die für die Fortbewegung keine von der Befehlsdecodierung und -ausführung unabhängige Routine aufweisen, und explizit ein GEH WESTLICH erfordern.
Wir wollen jedoch Richtungsanweisungen bevorzugt behandeln, was auch einer schnelleren Reaktion des Programmes auf die Eingabe hin zugute kommt.

Ersetzen wir zunächst die Zeile 1390, um adventuregemäße Eingaben zu ermöglichen:

```
1390 INPUT"Was soll ich tun";eingabe$:ei
ngabe$=UPPER$(eingabe$)
```

Bevor unser Programm nun irgendwelche Manipulationen des Spieles durchführt sollte es zunächst die Länge der Spielereingabe überprüfen. Ist die Eingabe länger als zwei Buchstaben (denn: oben = OB), wird es sich um irgendeine Handlung handeln, andernfalls muß die Bewegungsroutine durchlaufen werden. Diese testet zunächst, ob der Weg in der gewünschten Richtung überhaupt frei ist, wenn ja, wird in der Richtungstabelle unter der entsprechenden Richtung (Spalte) des Raumes, in welchem der Spieler sich aufhält (Reihe), der neue Raum abgelesen und der entsprechenden Variablen (spieler) zugewiesen. Abschließend wird der Spieler über die Durchführung der Aktion informiert:

```
1080 PRINT
1400 IF LEN(eingabe$)>2 THEN 1500
1410 IF eingabe$="N" AND durchgang(spiel
er,1)<>0 THEN spieler=durchgang(spieler,
1):PRINT"O.k.":GOTO 1080
1420 IF eingabe$="S" AND durchgang(spiel
er,2)<>0 THEN spieler=durchgang(spieler,
2):PRINT"O.k.":GOTO 1080
1430 IF eingabe$="W" AND durchgang(spiel
er,3)<>0 THEN spieler=durchgang(spieler,
3):PRINT"O.k.":GOTO 1080
1440 IF eingabe$="O" AND durchgang(spiel
er,4)<>0 THEN spieler=durchgang(spieler,
4):PRINT"O.k.":GOTO 1080
1450 IF eingabe$="OB" AND durchgang(spie
ler,5)<>0 THEN spieler=durchgang(spieler
,5):PRINT"O.k.":GOTO 1080
1460 IF eingabe$="U" AND durchgang(spiel
er,6)<>0 THEN spieler=durchgang(spieler,
6):PRINT"O.k.":GOTO 1080
1470 PRINT"DAHIN FUEHRT KEIN WEG !":GOTO
 1080
1500 REM ab hier spaeter Eingabeanalyse
```

Mit Beginn eines jeden Spieles müssen wir dem Programm natürlich den Startraum mitteilen:

```
150 spieler=1
```

Ein kurzer Spaziergang durch unsere Adventurewelt wird uns nun schnell davon überzeugen, daß wir als nächstes unbedingt die Gestaltung des Monitorbildes in Angriff nehmen müssen, oder unseren Augen wird sich nach einigen Spielzügen ein wenig informatives Durcheinander bieten.

FORMATIERUNG DER AUSGABE

Um eine saubere Bildschirmgestaltung zu ermöglichen, muß der Bildschirm bei Spielstart natürlich gelöscht werden:

```
1070 CLS
1080 PRINT:PRINT
```

Diese Leerzeilen sind auf einem leeren Schirm selbstverständlich sinnlos. Erforderlich werden sie immer erst nach Ausführung eines Befehles, denn die Eingabe des Spielers, wie auch eventuell erfolgte Mitteilungen an diesen, müssen um eine Zeile nach oben rutschen. Dieses Scrollen erzielen wir durch ein PRINT in die letzte Bildschirmzeile und ist als Abschluß eines jeden Spielzuges notwendig.
Erst mit der zweiten Print-Anweisung erzeugen wir eine Leerzeile, welche der Übersichtlichkeit des Bildes dient.

Uns stellt sich nun die Frage, wie wir nach Ausgabe der Informationen an den Spieler aus dem oberen Drittel des Schirmes in die unterste Zeile gelangen, denn schließlich wollen wir nicht alle dazwischenliegenden Zeilen überschreiben.

EXKURS: BILDSCHIRMADRESSIERUNG

Von den Fähigkeiten her bietet uns das Locomotive Basic des Schneider CPC zwei Möglichkeiten.
Zum einen wären da die Cursor - Steuerbefehle, mit denen die Schreibmarke um eine entsprechende Anzahl von Bildschirmpositionen in alle vier Richtungen und somit auch nach unten bewegt werden kann. Dazu muß jedoch die Anzahl der zu überspringenden Zeilen bekannt sein; diese wird jedoch bei jedem Spielzug, da von der Menge der

ausgegebenen Informationen, sprich Anzahl von Gegenständen, abhängig, variieren.
Ohne umfangreiche Kalkulationen wird dieser Weg daher nicht begehbar sein !

Da wir jedoch später einen dieser Steuerbefehle verwenden werden, soll an dieser Stelle auch kurz auf die Steuercodes eingegangen werden.

Vereinfacht kann gesagt werden, daß für den Basic - Interpreter jeder Befehl als eine Codezahl, als sogenanntes **TOKEN**, *und nicht als Buchstabenfolge existiert.*
Alle Routinen des Interpreters, die für die Bildschirmverwaltung vorgesehen sind, können unter Zuhilfenahme der Funktion CHR$() über diese Zahl angesprochen werden, wobei für die einfachen Cursorsteuerbefehle folgende Zuordnung gilt:

CHR$(8)	*Cursor links*
CHR$(9)	*Cursor rechts*
CHR$(10)	*Cursor nach unten*
CHR$(11)	*Cursor nach oben*

So würde beispielsweise die Anweisung PRINT CHR$(11) den Cursor um eine Zeile nach oben bewegen.

Für die Lösung unseres wie auch fast aller anderen sich bei der Schreibstellenadressierung stellenden Probleme ist der 'LOCATE'- Befehl, welcher den Cursor auf eine genau definierte Position setzt, weitaus besser geeignet.
Verwendet wird er in der Form LOCATE X,Y, wobei Y der Zeile und X der Schreibposition innerhalb dieser Zeile entspricht:

```
1390 LOACTE 1,25:INPUT"Was soll ich tun"
;eingabe$:eingabe$=UPPER$(eingabe$)
```

Unsere Spielanweisungen werden wir somit in Zeile 25 eingeben und das abschließende RETURN wird den gesamten Bildschirminhalt um eine Zeile nach oben schieben, weshalb für die Mitteilung der Reaktionen an den Spieler keine weiteren Maßnahmen getroffen werden müssen. Mit Beginn des neuen Zuges macht Zeile 1080 die unterste Reihe für den nächsten Eingabezyklus frei.

Leider werden beim derzeitigen Entwicklungsstand unseres Programmes alle Ausgaben in der untersten Bildschirmzeile gemacht werden, ein Effekt, der für die Ausgabe der Raumbeschreibungen keineswegs erwünscht war:

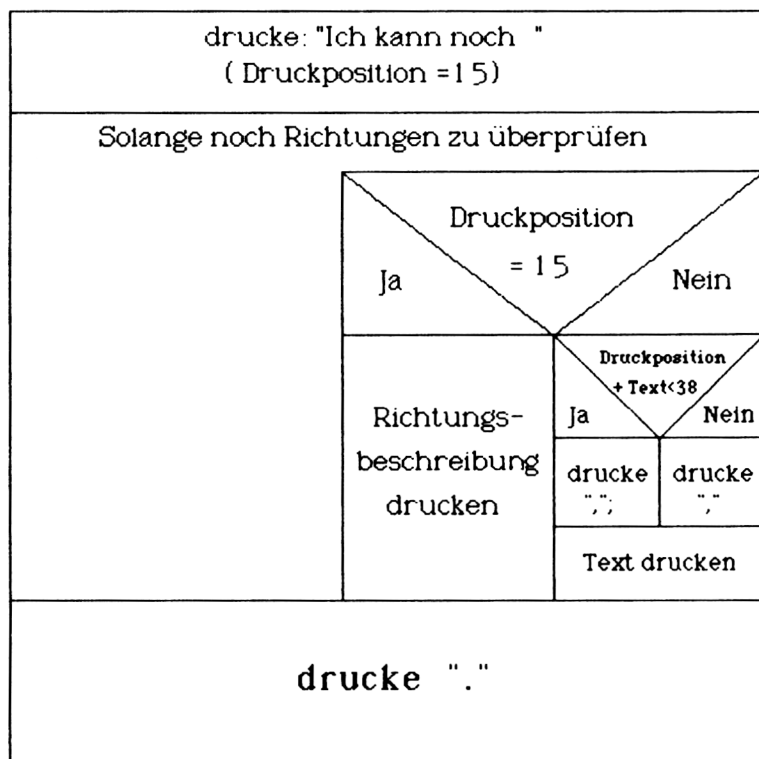
```
1130 LOCATE 1,1
```

Den Abenteurer über die freien Ausgänge zu informieren, wird allerdings nicht ganz so einfach sein, denn unsere bisherige Verfahrensweise läßt es nicht zu, mehr als drei Wege einwandfrei auszugeben.
Denn es sollte kein Wort über das Ende der Zeile hinausgehen, ebenso sollten die Bezeichnungen der verschiedenen Himmelsrichtungen durch ein Komma getrennt und der Satz mit einem Punkt abgeschlossen werden.

Entwickeln wir daher unter Zuhilfenahme des folgenden Beispieles eine Routine, welche diese Aufgabenstellung nach nebenstehendem Schema bewältigt.

```
Ich kann nach Norden, Sueden, Osten,
Oben, Unten.
```

Fest steht zunächst, daß bevor überhaupt irgendeine Richtung ausgedruckt werden kann, *Ich kann nach* auf dem Monitor zu erscheinen hat (1240).


drucke: "Ich kann noch "
(Druckposition = 1 5)
Solange noch Richtungen zu überprüfen
Druckposition
= 1 5
Ja
Nein
Richtungs-
beschreibung
drucken
Druckposition
+ Text<38
Ja
Nein
drucke
",";
drucke
","
Text drucken
drucke "."

Um nun innerhalb der Schleife die verschiedenen Möglichkeiten testen zu können, müssen wir zunächst Zeile 1260 ändern:

```
1260 IF durchgang(spieler,richtung)=0 TH
HEN GOTO 1310
```

Der erste Ausgang wird uns dann keine Schwierigkeiten machen, er kann ohne alle Vorarbeit ausgedruckt werden. Wir müssen diese erste Ausgabe nur von allen weiteren unterscheiden können, was auch ohne weiteres möglich ist. So wird die erste Richtungsangabe immer nach 'Ich kann nach ' an der fünfzehnten Schreibposition erscheinen, deshalb:

```
1270 IF POS(#0)=15 THEN PRINT richtung$(
richtung);:GOTO 1310
```

Vor der Ausgabe aller folgenden Richtungsangaben müssen wir uns zunächst jedoch vergewissern, daß der Text auch noch in die gerade zu beschreibende Zeile paßt. Sollte dies der Fall sein, drucken wir zunächst ein Komma und dann die betreffende Himmelsrichtung:

```
1280 IF POS(#0)+LEN(richtung$(richtung))
<38 THEN PRINT", ";richtung$(richtung);:
GOTO 1310
```

Sollte die Gesamtausgabelänge größer als 38 werden, erfolgt die Ausgabe eines Kommas sowie die Ansteuerung der nächsten Zeile mittels eines PRINT- Befehles:

```
1290 PRINT",":PRINT richtung$(richtung);
:GOTO 1310
```

Mit diesen drei Zeilen haben wir alle auftretenden Möglichkeiten abgedeckt. Der abschließende Punkt ist auf jeden Fall nur ein einziges Mal erforderlich und wird deshalb erst nach Verlassen der Schleife gesetzt:

```
1300 NEXT richtung
1320 PRINT"."
```

Vielleicht bedarf es noch einer Erklärung, warum wir die Zeilenlänge nur auf kleiner 38 testen, wo eine Bildschirmzeile doch von 1 bis 40 geht. Nun, die Antwort finden Sie sowohl in Zeile 1320 als auch im Komma in 1280.

Zu guter Letzt müssen wir uns über eine weitere Möglichkeit Gedanken machen, die nicht nur theoretisch, sondern manchmal, bei schwierigeren Adventurespielen fast immer, in der Praxis Verwendung findet.
Ich spreche von Räumen, die scheinbar keinerlei Verbindung zur übrigen Welt haben, so daß auch keine Himmelsrichtungen ausgegeben werden.
In solch einem Falle wäre das Bild auf dem Monitor, beim derzeitigen Stand unseres Programmes, recht unschön.
Ich kann nach wird zwar ausgegeben, aber dann folgt ein Punkt, wobei *nirgendwo* wohl die korrektere Lösung wäre.
Wir müssen uns daher, bevor unser Programm mit der nächsten Aktion fortfährt, vergewissern, ob schon irgendetwas ausgegeben wurde, was mittels einer weiteren Variablen, der Kontrollvariablen *gedruckt*, geschehen kann:

```
1240 PRINT"Ich kann nach ";:gedruckt=0

1260 IF durchgang(spieler,richtung)=0 TH
EN GOTO 1310 ELSE gedruckt=-1

1310 IF gedruckt=0 THEN PRINT"nirgendwo
 ";
```

Mit Start des Ausgabeteils wird *gedruckt* auf null gesetzt. Wird anschließend keine einzige Himmelsrichtung ausgegeben, so druckt Zeile 1310 das gewünschte ´nirgendwo´.

GOLD, SILBER UND ANDERE NÜTZLICHE DINGE

Eine Wanderung durch unsere Adventurewelt verschafft dem Abenteurer bislang einen recht trostlosen und uninteressanten Eindruck, stehen ihm doch keinerlei Handlungsgegenstände zur Verfügung. Darüberhinaus werden reine Ortsbeschreibungen beim besten Willen kein Spiel ermöglichen. Nehmen wir also wieder einmal unsere Karte zur Hand und bereichern wir unsere Welt um die gewünschten Gegenstände, wobei wir uns in der Regel auf für den Fortlauf des Spieles wichtige Objekte beschränken wollen. Denn für reine Verschönerungen wäre der spätere Aufwand zu groß, schließlich müssen wir gewährleisten, daß der Spieler alles das, was er findet, auch in die Hand nehmen und untersuchen kann. Dennoch kommen wir nicht umhin, in jedem Raum mindestens ein Objekt zu plazieren, denn daß der Abenteurer so ohne weiteres gar nichts sieht, ist doch sehr zweifelhaft. Falls die Phantasie nicht ausreicht, ist das betreffende Objekt eben *nichts besonderes*, eine Formulierung die wir auch deshalb so häufig in Adventurespielen finden, weil sie gerne zur Verminderung des Arbeitsaufwandes gewählt wird, denn in einem normalem Wald beispielsweise sind Bäume, Sträucher, Gras, Insekten usw. eben wirklich nichts besonderes.
Oftmals kann jedoch auch auf unnötige Dinge nicht verzichtet werden, besonders dann nicht, wenn sie zum Milieu gehören und das Adventure einen realistischen Eindruck hinterlassen soll.

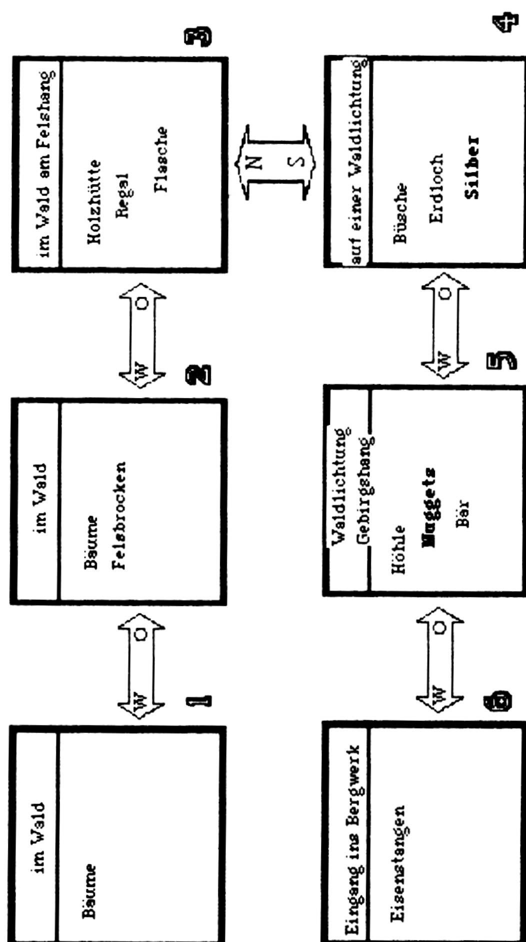
So werden auch wir in unserem Wald Bäume aufstellen müssen, obwohl die Handlung sie nicht erfordert. In Raum zwei, der bereits nahe dem Gebirge liegt, vertiefen einige Felsbrocken den Eindruck der Wirklichkeit. Weiterhin gehört zu einem Bergwerk eine Baracke, egal ob es nun die Wohnstätte des Minenbesitzers oder ein Geräteschuppen ist. Irgendwo müssen wir schließlich auch die zu suchenden Schätze verstecken, zusätzlich sind einige Gegenstände, die dem Spieler eher schaden als nützen, angebracht.

Lassen Sie mich daher bitte im folgenden den mir vorschwebenden Handlungsablauf näher konkretisieren und die benötigten Gegenstände auswählen:

HANDLUNGSABLAUF GOLDRAUSCH

Die Miniversion soll dem Spieler die Aufgabe stellen, zwei Schätze, Goldstücke und Silbermünzen, im Umfeld der Mine zu finden. Bei Spielbeginn soll sich der Spieler im Wald befinden und sich zunächst an das Bergwerk herantasten müssen. Auf dem Weg zur Mine entdeckt er im Wald ein Erdloch, auf dessen Grund sich eine schwere Eisentruhe befindet. In dieser befindet sich das Silber, doch leider ist die Truhe mittels Vorhängeschloß und einer schweren Eisenkette verschlossen; ein passender Schlüssel läßt sich nirgends finden. Doch am Eingangsstollen des Bergwerkes liegen einige Eisenstangen, die stabil genug sind, um damit die Kette zu zerreißen oder das Schloß zu sprengen. Wir können es den Spieler auch mit Sprengstoff versuchen lassen, soll er in einer Holzhütte ruhig Dynamit finden - für uns eine gute Gelegenheit, sein vorzeitiges Ende zu programmieren. Die Goldstücke deponieren wir in einer Felsenhöhle, die zum Leidwesen des Spielers von einem wilden Bären bewohnt wird. Das wird der Spieler jedoch erst erfahren, wenn er die Höhle untersucht, sollte die Gier nach dem gelben Metall siegen und ihn dazu verleiten, sofort zuzugreifen, wird ebenso der Hunger des Bären selbigen seine Scheu vor dem Menschen verlieren lassen - nach dem Dynamit die zweite Falle für den Abenteurer.

Andererseits wissen wir alle dank Wilhelm Busch, daß man Bären mit Honig eine große Freude machen kann,



stellen wir auf einem Regal in der Hütte daher eine Flasche bereit, die den begehrten Stoff enthält. Betritt der Spieler damit gewappnet die Höhle, wird der Bär den Honig wittern und mit der Flasche in den Tiefen der Höhle verschwinden, womit einem erfolgreichem Ende dieses Miniadventures nichts mehr im Wege steht.

Die Karte zu unserem Adventure *Goldrausch* dürfte nun ähnlich der vorangegangenen Abbildung aussehen.

DIE OBJEKTE

Bevor wir mit der Programmierung der Handlungsgegenstände unser Adventure vervollständigen, werden einige weitere Vorüberlegungen notwendig.

Um dem Spieler das Hineinleben in unsere Computerwelt zu ermöglichen, werden wir diese Welt nicht mit trockenen Substantiven beschreiben können, sondern wir werden uns bemühen, das Interesse des Spielers durch plastische Schilderungen wachzuhalten. Die lapidare Mitteilung ´Ich sehe einen Bären.´ läßt den Spieler unser Adventure zwar spielen, die Nachricht ´Ich sehe einen äußerst grimmig dreinblickenden Bären´ wird ihn unser Adventure dagegen erleben lassen.
Sollte in dem Abenteurer dann das Bestreben erwachen, einen neuen Freund zu finden, dann wäre eine Eingabe wie ´Streichel äußerst grimmig dreinblickenden Bären´ für uns als Programmierer zwar einfach zu realisieren, für den Spieler hingegen unzumutbar.
Daher werden wir jedem Objekt neben der eigentlichen Objektbeschreibung einen Rufnamen zuordnen, womit wir auch die Möglichkeit erhalten, die Wortlänge unseres Adventures nach Bedarf frei festzulegen.

So erfordern professionelle Adventures aus dem englischsprachigen Raum meist nur die ersten drei oder vier Buchstaben eines jeden Wortes zur Identifikation. Drei Buchstaben sind auch für deutsche Adventures in der Regel ausreichend, nur bei der Realisation eines größeren Projektes sollten Sie zuvor eine Liste der vorgesehenen Gegenstände erstellen und diese aufmerksam durchsehen, ansonsten werden Sie bei der Programmierung der Bedingungen und Aktionen überrascht feststellen, wie viele für ein Adventure wichtige Substantive mit Sch beginnen.

Als dritte Information neben dem ausführlichen und dem Kurznamen werden wir zu jedem Objekt die Nummer des Raumes speichern müssen, in dem der Gegenstand sich gerade befindet. Da sich diese Zahlen während des Spieles laufend ändern , können sie nicht aus einer fest programmierten Programmzeile abgelesen werden, sondern müssen in entsprechenden Variablen bereitstehen.
Erzeugen wir daher eine Liste, in der jeder Eintrag dem Aufenthaltort eines Objektes entspricht.
Zweckmäßigerweise verwenden wir die indizierte Variable OB() und geben uns damit ein einfaches Mittel zur Hand, um die Position eines jeden Gegenstandes kontrollieren und manipulieren zu können.
Denn solange ein Gegenstand nicht aktiv am Spiel beteiligt ist, soll OB = 0 sein, wir sagen, der Gegenstand befindet sich im *Lagerraum*. Andernfalls entspricht der Inhalt von OB() entweder dem Wert der Variablen Spieler und das betreffende Objekt ist somit im Raum sichtbar, oder aber er ist gleich -1 in dem Falle, daß der Abenteurer den Gegenstand mit sich führt (*Inventory*).

Die folgenden Zeilen stellen alle diese Informationen zu den Gegenständen, die für eine Realisation des bisherigen Spielplanes (Goldrausch, Miniversion) erforderlich sind, bereit.

Nach der ausführlichen Beschreibung folgt der Kurzname, welcher mit den Eingaben des Spielers verglichen wird, dann die Positionsnummer, die den betreffenden Gegenstand einem Raum zuordnet.
Die Anführungszeichen sind übrigens nur dann unbedingt erforderlich, wenn innerhalb der Beschreibung selbst wieder ein Komma verwandt wird.

```
300 REM ******************* GEGENSTAENDE
301 DATA"viele grosse Baeume","baeu",1
302 DATA"viele grosse Baeume","baeu",2
303 DATA"einige Felsbrocken","fels",2
304 DATA"eine verfallene Holzhuette","hu
et",3
305 DATA"eine verschmutzte Korbflasche",
"flas",0
306 DATA"Honig","honi",0
307 DATA"eine Holzkiste","kist",3
308 DATA"ein klappriges Regal","rega",0
309 DATA"etwas Sprengstoff","spre",0
310 DATA"ein duesteres Erdloch","erdl",0
311 DATA"eine rostige Eisentruhe","truh"
,0
312 DATA"*Silbermuenzen*","silb",0
313 DATA"eine duestere Felsenhoehle","ho
eh",5
314 DATA"einen grimmig dreinblickenden B
aeren","baer",0
315 DATA"zahllose niedrige Buesche","bue
s",4
316 DATA"mehrere Eisenstangen","eise",6
317 DATA"*Nuggets*","nugg",5
```

Genau wie bei den Raumbeschreibungen müssen wir vor Initialisierung der Variablen auch hier wieder die Anzahl der Datenelemente festlegen:

```
120 ao=17
830 FOR objekt=1 to ao
835 READ ob$(objekt),rufname$(objekt),ob
(objekt):rufname$(objekt)=UPPER$(rufname
$(objekt))
840 NEXT objekt
```

```
190 DIM raum$(ar),durchgang(ar,6),ob$(ao
),rufname$(ao),ob(ao)
```

Zur Speicherung aller Objekte sind 17 verschiedene Variablen notwendig, für die der benötigte Speicherplatz durch die DIMensionierungsanweisung in Zeile 190 bereitgestellt wird.
Wie die meisten Computer, so reserviert auch unser Schneider CPC mit dem Start von Basic elf Elemente pro Liste, weshalb wir trotz des Fehlens dieser Zeile bei der Initialisierung der Orte keine Fehlermeldung erhalten hatten.

Machen wir die Einrichtung eines Raumes nun dem Spieler sichtbar:
In der Variablen SPIELER finden wir die Nummer des betretenen Raumes, OB() enthält die Raumnummern aller Objekte. Überprüfen wir daher innerhalb einer Schleife, ob die Positionsnummer eines Gegenstandes gleich der aktuellen Raumnummer ist:

```
1160 PRINT"Ich sehe ";
1170 FOR i=1 TO ao
1180 IF ob(i)<>spieler THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+"<=39 THEN P
RINT ob$(i);", ";:GOTO 1210
1210 NEXT i
```

Programmzeile 1180 testet für jeden Gegenstand, ob dieser sich in einem anderen Raum als der Spieler befindet, fällt dieser Test positiv aus, wird sofort das nächste Objekt

überprüft. Ansonsten wird die betreffende Objektbeschreibung in Zeile 1190 ausgegeben. Das Semikolon sorgt dafür, daß nicht für jedes weitere Objekt eine neue Ausgabezeile benutzt wird. Zur Vermeidung einer Ausgabe über die Zeile hinaus ergänzen wir:

```
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT : GOTO 1190
```

Vor Ausgabe des nächsten Gegenstandes wird die Gesamtlänge des Ausdrucks errechnet. Wenn der Cursor durch Drucken der Gegenstandsbeschreibung, des Kommas und der Leerstelle zwischen zwei Objekten (daher +2) eine Position größer Spalte 39 erreichen würde, wird ein Zeilenvorschub ausgelöst. Danach verzweigt das Programm wieder zur Zeile 1190, die Bedingung ist nun erfüllt und die Beschreibung wird ausgedruckt.
Hinter dem letzten Objekt wirkt das Komma jedoch unschön, weshalb wir den Cursor um zwei Positionen nach links bewegen, und einen Punkt über das Komma schreiben:

```
1220 PRINT CHR$(8);CHR$(8);"."
```

Zum Abschluß betrachten wir auch hier wieder die Möglichkeit, daß keine Daten zur Ausgabe an den Spieler vorliegen und der Satz *Ich sehe* mit *nichts besonderes* ergänzt werden muß.
Das Prinzip ist inzwischen bekannt, so daß die notwendigen Ergänzungen an dieser Stelle ohne weitere Kommentare gelistet werden können:

```
1160 PRINT"Ich sehe ";:gedruckt=0
1190 IF POS(... ... :GOTO 1205
 1205 gedruckt=-1
1215 IF NOT gedruckt THEN PRINT"nichts b
esonderes  "
```

Nun sind wir unserem gesteckten Ziel schon recht nahe, was uns noch fehlt, um den Bildschirmaufbau der amerikanischen Originaladventures zu erreichen, sind eine Trennungslinie sowie eine weitere Leerzeile:

```
1010 leerzeile$=STRING$(40," ")
1230 PRINT leerzeile$;
1330 PRINT"-----------------------------
----------"
```

Ein letztes Problem werden wir wiederum erst nach mehreren Eingaben feststellen. Die Mitteilungen innerhalb der unteren Schirmhälfte rücken immer weiter nach oben und vermischen sich nach einigen Zügen mit den Beschreibungen der Umgebung. Es muß daher Aufgabe des Programmes sein, vor deren Ausgabe die obersten Bildschirmzeilen zu löschen, was durch Drucken einer ausreichenden Menge von Leerzeilen geschehen kann:

```
1090 LOCATE 1,1
1100 FOR zeile=1 TO 10
1110 PRINT leerzeile$
1120 NEXT zeile
```

DER WORTSCHATZ

Bevor wir nun das eigentliche Spiel programmieren und uns mit den für einen Spieler nicht sichtbaren Aktionen des Programmes beschäftigen, müssen wir schnell noch die dazu notwendigen Verben eingeben. Diese werden, zusammen mit den zuvor erwähnten Rufnamen der Objekte, den kompletten Wortschatz unseres Adventures darstellen und somit Grundlage zur Verständigung mit dem Spieler sein.
Neben einer Reihe von immer erforderlichen Verben wie *Untersuche*, *Nimm*, *Leg* werden auch sie vom Inhalt des

jeweiligen Adventures abhängen. Die Implementation erfolgt entsprechend den Räumen und Objekten, wobei eine der Wortlänge des Adventures entsprechende Abkürzung ausreichend ist:

```
130 av=6
190 DIM raum$(ar),durchgang(ar,6),ob$(ao
),rufname$(ao),ob(ao),verb$(av)

200 REM ************************** VERBEN
201 DATA unte,nimm,leg ,oeff,benu,zers

810 FOR i=1 TO av
815 READ verb$(i):verb$(i)=UPPER$(verb$(
I))
820 NEXT i
```

Weiterhin können wir mit einigen wenigen Ergänzungen das Bild attraktiver gestalten. Unterschiedliche Farben werden uns helfen, die während des Spieles vielfältigen Mitteilungen zu differenzieren. Ich habe mich für einen schwarzen Hintergrund, auf dem die Beschreibungen der Welt in dunklem Blau an uns ausgegeben werden, entschieden. Die freien Wege wie auch die Mitteilungen an uns werden in hellem Blau ausgedruckt; unsere Eingaben werden in weiß gemacht.
Selbstverständlich steht es Ihnen frei, andere Farben, entsprechend Ihrem Geschmack, zu verwenden.
Dazu müssen Sie nur den zweiten Wert bei den INK - Befehlen in Zeile 10 ändern.

```
10 PAPER 0:INK 0,0:BORDER 0:PEN 1:INK 1,
2:PEN 2:INK 2,14:PEN 3:INK 3,26

1130 LOCATE 1,1:PEN 1
1230 PRINT leerzeile$:PEN 2
1290 PEN 3:LOCATE 1,25:INPUT"Was soll ic
```

```
h tun";eingabe$:eingabe$=UPPER$(eingabe$
):PEN 1
```

Damit haben wir die im ersten Kapitel dieses Buches beschriebene äußere Gestaltung erreicht und können uns dem Spielverhalten unseres Programmes widmen.

ANALYSE DER EINGABEN

Vom Spieler gewünschte Richtungsänderungen, eine Befehlsgruppe mit Wortlängen von ein bis zwei Buchstaben, werden bereits erkannt und korrekt ausgeführt. Die Standardeingabe besteht jedoch immer aus Verb und Objekt, wobei die Wortlängen, abgesehen von der zur Erkennung der Begriffe notwendigen Mindestwortlänge, keinen Restriktionen unterworfen werden. Um wie vom Spieler gewünscht reagieren zu können, muß unser Programm dessen Eingabe überprüfen:

ist das benutzte Verb programmiert ?
ist das Objekt vorgesehen ?

Können beide Fragen mit ja beantwortet werden, wird die Eingabe in Verb und Objekt zerlegt, Verb- und Objektnummer werden ermittelt; sind die Bestandteile der Eingabe nicht definiert, wird eine entsprechende Mitteilung an den Abenteurer ausgegeben.

EXKURS: STRINGBEHANDLUNG

Der wesentliche Unterschied zwischen einem Computer und einem programmierbaren Taschenrechner ist die Fähigkeit des Computers, mit Texten umzugehen. Natürlich hat ein Rechner bislang kein Verständnis für diese Texte, sondern er faßt sie als eine Ansammlung bestimmter Zeichen, als sogenannte Zeichenkette oder String auf. Bei diesen Strings handelt es sich im allgemeinen um Kombinationen aller per Tastatur eingebbarer Zeichen. Das Betriebssystem des Computers stellt nun eine Reihe von Funktionen zur Behandlung dieser Zeichenketten zur Verfügung.
Neben einer Reihe von Konvertierungsbefehlen, zu denen neben UPPER$ und LOWER$ auch die Funktionen VAL() und STR$(), welche eine Zahl in einen String bzw. einen String

in eine Zahl verwandeln, unter anderem die Befehle LEFT$, RIGHT$ und MID$.
Diese drei Funktionen stellen dem Programm zur weiteren Bearbeitung einen genau definierten Teilstring aus einer anderen Zeichenkette zur Verfügung. So ergibt die Anweisung LEFT$("ABCDEF",3) eine Kette der linken drei Zeichen, also den Substring ABC.
Als notwendige Parameter müssen somit die zu bearbeitende Zeichenkette wie auch die Anzahl der gewünschten Buchstaben bekannt sein. Entsprechendes gilt für RIGHT$.
MID$(X$,S,X) stellt uns dagegen einen ab Position S beginnenden, X Zeichen langen Teilstring aus X$ bereit.
So wird unser Adventureprogramm beispielsweise die Position des Leerzeichens zwischen Verb und Objekt der Eingabe eines Spielers ermitteln und unter Verwendung dieser Kennzahl mittels der Funktionen LEFT$ und RIGHT$ aus EINGABE$ die zwei Substrings EVERB$ und EOBJEKT$ ermitteln.
Um die Anzahl erforderlicher Buchstaben berechnen zu können, wird zuvor mit LEN(EINGABE$) die Gesamtlänge der Spielanweisung ermittelt.

Wird die Ausführung eines Spielzuges nicht in den Zeilen 1410 bis 1470 durchgeführt, so wird, falls die Länge der Eingabe weniger als drei Buchstaben beträgt, angenommen, daß es sich um einen Eingabefehler handelt. Nachdem *Dahin führt kein Weg!* ausgegeben worden ist, beginnt ein neuer Zug.
Handelt es sich um eine längere Eingabe, wird der Programmlauf mit Zeile 2000 fortgesetzt:

```
1470 IF LEN(eingabe$)<3 THEN PRINT"Dahin
 fuehrt kein Weg !":GOTO 1080

2000 laenge=LEN(eingabe$)
2010 FOR buchstabe=1 TO laenge
2020 pruef$=MID$(eingabe$,buchstabe,1)
2030 IF pruef$<>" "THEN NEXT buchstabe
```

```
160 wortlaenge=4
2040 everb$=LEFT$(eingabe$,wortlaenge)
```

Nach Ermittlung der Länge der gesamten Eingabe wird innerhalb einer Schleife jeder Buchstabe, beginnend von links, darauf überprüft, ob er mit einem Leerzeichen identisch ist. Ist das Leerzeichen gefunden, steht die Länge des eingegebenen Verbes (vom ersten bis zum gerade geprüften Zeichen minus eins) fest.
Dieser Wert, abgezogen von der Gesamtlänge der Eingabe, entspricht der Anzahl der Buchstaben des eingegebenen Objektes.
Somit können EVERB$ und EOBJEKT$ unter zusätzlicher Berücksichtigung der spieltypischen Wortlänge ermittelt werden.

Wurde ein ´Ein Wort Befehl´ verwendet (HELP), so wird die Verblänge gleich der Eingabelänge sein. Da kein Objekt zur weiteren Bearbeitung bereitgestellt werden muß, wird direkt (Zeile 2060) zur Verbanalyse (Zeile 2090) verzweigt.

```
2040 everb$=LEFT$(eingabe$,wortlaenge)
2050 rl=laenge-buchstabe
2060 IF rl<0 THEN 2090
2070 eobjekt$=RIGHT$(eingabe$,rl)
2080 eobjekt$=LEFT$(eobjekt$,wortlaenge)
2090 FOR verbnummer=1 TO AV
2100 IF everb$=verb$(verbnummer) THEN 21
30
2110 NEXT verbnummer
```

Innerhalb einer weiteren Schleife wird das erste der eingegebenen Worte mit den im Spiel möglichen Verben verglichen. Wird ein identischer String gefunden, dann entspricht der Schleifenindex der Verbnummer und die Schleife wird verlassen.

Wurde die Schleife vollständig abgearbeitet und keine Identität festgestellt, so hatte der Programmierer das betreffende Verb nicht vorgesehen, und dem Spieler wird darüber Mitteilung gemacht:

```
2120 PRINT"Das Verb verstehe ich nicht "
":GOTO 1080
```

Anschließend wird nach gleichem Prinzip die Objektnummer ermittelt:

```
2130 FOR o=1 TO ao
2140 IF eobjekt$=rufname$(o) THEN 2200
2150 NEXT o
2160 PRINT"Ich verstehe das Objekt nicht
 !":GOTO 1080
```

Mit Eingabe dieser Zeilen haben wir einen wesentlichen Teil unserer Entwicklungsarbeit hinter uns gebracht. Die bislang entwickelten Routinen sorgen für einen ansprechenden Aufbau des Bildschirmes, wie sie auch das ´Verständnis´ für die Eingaben des Spielers erzeugen.
Sie ermitteln, welches der möglichen Verben der Abenteurer benutzt hat und mit welchem Gegenstand er etwas tun möchte, womit eine Verzweigung des Programmablaufes an eine zur Ausführung der betreffenden Aktion vorgesehene Programmzeile möglich wird. Wegen dieser zentralen Steuerungsaufgaben wird dieser Programmteil auch als TREIBER bezeichnet.

ÜBERBLICK: AUFBAU DER PROGRAMME

Somit wird deutlich, daß unsere Abenteuerprogramme im wesentlichen aus drei Teilen bestehen:

1. *Daten des Adventures*
2. *Adventuretreiber*
3. *Ausführung der Spielzüge*

1. Die Daten stellen die Grundlage eines jeden Spieles dar. Sie werden zum Teil an Arbeitsvariablen übergeben oder innerhalb von kurzen Unterprogrammen für die Übergabe an das Hauptprogramm bereitgestellt.

2. Dem Treiber fällt die Steuerung des gesamten Programmablaufes zu. Er gestaltet die Bildschirmausgabe, übernimmt die Eingaben des Spielers, wertet sie aus und leitet die Befehlsausführung ein.
Der Treiber ist vom jeweiligen Adventure unabhängig und kann unverändert für alle Adventures übernommen werden.

3. Der dritte Programmteil stellt das eigentliche Adventure dar. Treffen alle für eine Handlung notwendigen Bedingungen zu, so wird diese Aktion ausgeführt.

Damit Ihnen der Überblick erhalten bleibt, fassen wir an dieser Stelle den Aufbau unserer Adventures zusammen, wobei bereits auch die noch im folgenden zu erstellenden Routinen mit angegeben werden.
Aus dieser Zusammenstellung geht bereits hervor, wie universell unser Konzept eingesetzt werden kann.
So werden einige kleine Änderungen am Treiber ausreichen, um aus einem Textadventure ein Grafikadventure zu machen, ebenso wird es, Dank des einheitlichen, systematischen Aufbaus mit genau definierten Programmzeilen für die diversen Funktionsblöcke, möglich sein, einen Adventuregenerator zu entwickeln.

KONZEPTIONELLER AUFBAU: TEXTADVENTURE

DATEN

0	100	TITELBILD
100	200	KONTROLLDATEN
		SPEICHERPLATZ RESERVIEREN
200	300	VERBEN
300	500	OBJEKTBESCHREIBUNGEN
500	600	RAUMBESCHREIBUNGEN
		VERBINDUNGEN DER RÄUME
600	700	STANDARDMITTEILUNGEN
700	800	EVTL. 2. TITEL
800	900	SPIELDATEN EINLESEN
900	1000	ALLGEMEINE SPIELANWEISUNG

ADVENTURETREIBER

1000	1070	TREIBER INITIALISIEREN
1080	1330	BILDSCHIRMVERWALTUNG
1331	1389	BES. EREIGNISSE
1390		EINGABE DES SPIELZUGES
1391	1399	FALLEN & HINDERNISSE
1400	1500	HAUPTPERSON BEWEGEN
1500	1600	INVENTUR
1600	1700	SAVE GAME
1700	1800	LOAD GAME
1800	1950	HELP, VOKABULAR, INSTRUKT.
1950	2000	SPIELABBRUCH
2000	2160	ANALYSE DER EINGABE
2200		SPRUNGTABELLE AUSFÜHRUNG

AUSFÜHRUNG

5000	30000	SPIELZÜGE

WANN GEHT WAS ?

Stellen wir uns nun den Spieler vor, der unsere zwei Schätze finden will. Mit einem Blick stellt er fest, daß er sich mitten in einem Wald befindet, umgeben von Bäumen und Felsbrocken. Als geübter Adventurespieler oder Leser der vorangegangenen Kapitel, weiß er, daß des Pudels Kern in den unscheinbarsten, alltäglichsten Dingen liegen kann.

Wie wird er reagieren ?
Welche Eingaben müssen wir erwarten ?

Mit an Sicherheit grenzender Wahrscheinlichkeit probiert er mit Anweisungen wie *UNTERSUCHE WALD, UNTERSUCHE BAUM, oder UNTERSUCHE FELSBROCKEN* weiterzukommen.
NIMM BAUM ist weniger wahrscheinlich, dennoch müssen wir, um allen Kritikern unserer Programme den Wind aus den Segeln zu nehmen, eine Mitteilung der Art *So stark bin ich nicht.* programmieren, außerdem animieren vielfältige Mitteilungen, die zeigen, daß eine Eingabe auch wirklich verstanden wurde, den Abenteurer zum Weiterspielen.
Verständlich - wäre das Ergebnis jeder zweiten oder dritten Eingabe ein *Ich verstehe nicht was Du meinst !* dann wird der Reiz zum Abschalten des Computers doch sehr groß.

In der Praxis werden wir für die Räume 1 und 2 daher, weil wir hier noch keine Handlung vorgesehen haben, sondern nur unsere Welt größer wirken lassen wollen, nur die Ausgabe diverser Meldungen an den Spieler vorsehen, eine Aufgabe, die ein Print - Befehl schnell erledigt. Doch nachdem der Spieler Raum 3 betreten und die Holzhütte entdeckt hat, wird er diese ebenfalls in Augenschein nehmen und dabei das Regal mit der Korbflasche entdecken. Diese muß natürlich ab diesem Zeitpunkt, zusätzlich zum Regal und zur Hütte, in der Beschreibung der Szene erscheinen ('Ich sehe ...'), das heißt, wir müssen ihren Aufenthaltsort verändern. Weiteres

Kopfzerbrechen könnte uns die Eingabe *NIMM FLASCHE* bereiten: nun muß die soeben erschienene Korbflasche wieder verschwinden. Zurück an ihren alten Platz kann sie nicht, denn wenn unser Spieler *INVENTUR* macht, muß sie selbstverständlich aufgelistet werden.

Ebenso selbstverständlich kann der Spieler die Flasche nicht schon in Raum 1 untersuchen, genau so wenig in Raum 2, überhaupt sollten alle Eingaben betreffend der Flasche abschlägig beantwortet werden, solange diese sich nicht in unmittelbarer Nähe des Spielers befindet.

Dies ist jedoch eine Geschmacksache, die im Ermessen des Programmierers liegt. Denn selbstverständlich darf er die Meinung vertreten, daß die Flasche in Wirklichkeit bereits seit undenklichen Zeiten unberührt in dem Regal steht, und daß der Spieler sie nur noch nicht gesehen hat.
Wird dieser Spieler durch die zahlreichen Überraschungen eines Abenteuers jedoch gezwungen, noch einmal von vorne zu beginnen, kennt er die Startpositionen diverser Gegenstände, und man kann es ihm ohne weiteres ermöglichen, diese zu nehmen, auch ohne daß sie in der Zeile ´Ich sehe ...´ gelistet worden waren.

In der Tat wird der Vorgang des Nehmens dadurch sogar einfacher zu programmieren, schließlich muß eine Bedingung weniger überprüft werden, doch wollen wir es wie die Mehrheit der Adventureproduzenten halten und nur sichtbare Gegenstände auch greifbar sein lassen.

So wird denn auch die Holzhütte von unserer Hauptperson nur untersucht werden können, wenn sie sich in greifbarer Nähe befindet. Den Honig werden wir erst entdecken, nachdem wir die Korbflasche in unsere Hände genommen und geöffnet haben. Der Bär wird sich nur friedlich verhalten, wenn er durch den Verzehr des Honigs anderweitig beschäftigt ist.

Alle diese Bedingungen müssen vor Ausführung eines Befehles zunächst auf ihre Richtigkeit überprüft werden. Bereits unser kurzes Adventure mit nur sechs Räumen wird somit eine Vielzahl an Programmzeilen erfordern. Berücksichtigen wir nur die zu diesem Zeitpunkt der Entwicklung eingebrachten Verben untersuche, nimm, lege weg, öffne, benutze, zerstöre und unsere siebzehn Gegenstände, so müssen wir bereits Aktionen für 102 unterschiedliche Spielereingaben programmieren. Würden wir die Programmzeilen alle ohne besonderes Konzept hintereinanderschreiben, wäre es leicht, für die auftretenden, recht langen Ausführungszeiten des Programmes eine Erklärung zu finden. Um sie jedoch möglichst gering zu halten werden wir den nun zu erstellenden Programmteil in mehrere Blöcke aufteilen.

Prinzipiell stehen uns zwei voneinander verschiedene Möglichkeiten zur Verfügung. Zum einen könnten wir für jeden Raum eine ausreichende Anzahl von Programmzeilen reservieren, und dort für jede mögliche Aktion des Spielers ein paar Statements reservieren. Diese Technik wird recht häufig gewählt, hat auch den Vorteil, daß sie wegen der direkten Ausführung eines Befehles, der ja zuvor nicht irgendwie aufbereitet werden muß, recht schnell ist, und daß vor allem ein recht übersichtliches Programm entsteht, welches bequem und einfach, ohne Nachsehen in irgendwelchen Listen, erweitert oder edititiert werden kann. Sollten Sie sich einmal mit dieser Ausführung eines Adventureprogrammes beschäftigen wollen, dann soll die folgende Realisation unseres Raumes 1 als Anregung genügen. Beachten Sie, daß es sich um ein Beispiel handelt, welches mit unserem Konzept nichts zu tun hat, überschreiben Sie daher bitte keine Zeilen unseres bisher erstellten Programmes.

```
100 INPUT"Was soll ich tun";eingabe$
200 REM eingabe$ in verb$ und objekt$
zerlegen (wie besprochen)
300 ON raum GOTO 1000, 2000, 3000, 4000,
```

```
 5000, 6000
301 REM abhaengig von der Raumnummer in
den betreffenden Raum springen

1000 CLS:REM ******************** Raum 1
1010 PRINT"Ich bin im Wald."
1020 PRINT"Ich sehe viele grosse Baeume.
"
1030:
1040 REM weitere Gegenstaende ausdrucken
1100 PRINT"Ich kann nach Sueden, Osten."
1200 IF eingabe$="S" THEN raum=6: GOTO 1
00
1210 IF eingabe$="O" THEN raum=2: GOTO 1
00
1260 IF LEN(eingabe$)<3 THEN PRINT "Dahi
n fuehrt kein Weg !": GOTO 100
1300 IF eingabe$="INV" OR "INVENTUR" THE
N GOTO 200
1400 IF everb$="UNTE" AND eobjekt$="BAEU
" THEN PRINT"Ich sehe nichts besonderes.
":GOTO 100
1410 IF everb$="NIMM" AND eobjekt$="BAEU
" THEN PRINT"So stark bin ich nicht.":GO
TO 100
1999 PRINT"Ich verstehe nicht, was Du me
inst.": GOTO 100
2000 ab hier Behandlung Raum 2
3000 ab hier Behandlung Raum 3
```

Wenn Sie sich obige Zeilen angesehen oder sogar in Ihren CPC eingegeben haben, werden Sie feststellen, daß auf eine noch einfachere Art ebenfalls ein funktionsfähiges Adventure entstehen kann, und Sie werden sich vielleicht fragen, warum wir nicht dieses Konzept verwendet haben, wo es zudem auch noch ohne lange Kommentare zu verstehen ist ?

Nun, einerseits werden besonders umfangreiche Adventures weitaus mehr Speicherplatz benötigen, denn wir würden es nicht vermeiden können, identische Routinen mehrmals aufzuführen, da wir, um bei einem bekannten Beispiel zu bleiben, die Flasche in jedem Raum ablegen (sechs identische Programmzeilen) und, nicht zu vergessen, auch wieder nehmen können müssen. Diesen zwölf Zeilen stehen in unserem Adventuresystem ganze zwei Programmzeilen gegenüber. Allein das rechtfertigt unsere Wahl, doch auch das Argument der Übersichtlichkeit und Editierfreundlichkeit ist schnell zu entkräften.
Stellen sie sich vor, bei der Erweiterung des Adventures werden zur Fortführung der Handlung weitere Flaschen benötigt, so daß wir aus diesem oder irgendwelchen anderen Gründen aus der Flasche lieber ein kleines Holzfaß machen wollen. Hatte Ihr Adventure zu diesem Zeitpunkt bereits vierzig Räume, dürfen Sie nun in entsprechend vielen Zeilen aus ´Nimm Flasche´ ein ´Nimm Faß´ machen, entsprechendes gilt für ´Leg Flasche, Oeffne Flasche´ usw.
Haben Sie sich hingegen an unser Adventuresystem gehalten, ändern Sie Zeile 305, fertig !

Vergessen wir daher diese Methode und wenden wir uns wieder unserem Adventure zu. Wir lassen die Handlung ebenfalls in genau definierten Programmzeilen ablaufen, doch enthält nun jeder Block die Behandlung jeweils eines Verbes.

Unser Treiber hat bereits Verb- und Objektnummer ermittelt, daher können wir diese Blöcke gezielt anspringen. So ist es sinnvoll, allen Zeilen, die mit der Ausführung des Befehles ´untersuche´ beschäftigt sind, beispielsweise Programmzeilen ab 5000 zuzuteilen, die Aktion ´nim´ wird dann entsprechend ab Zeile 6000 realisiert.

Der Steuerung des Programmablaufes nimmt der Adventuretreiber anhand der Verbnummer in Zeile 2200 vor:

```
2190 REM                    UNTE NIMM LEG
OEFFNE BENUTZE ZERSTOERE
2200 ON verbnummer GOTO 5000,6000,7000,8
000,9000,10000
```

Mit Programmzeile 5000 beginnt nun der Block 'UNTERSUCHE'. Es ist leicht einzusehen, daß uns hier der größte Arbeitsaufwand abverlangt wird.
Nicht nur, daß auf *UNTERSUCHE HÜTTE* die Nachricht *IN DER HUETTE STEHT EIN ALTES REGAL.* ausgegeben und das Regal als sichtbares Objekt erscheinen muß. Nein, wir müssen auch sicherstellen, daß die Hütte an dem gerade besuchten Ort zur Verfügung steht, und gegebenenfalls eine Mitteilung wie *SO ETWAS SEHE ICH HIER NICHT.* an den Spieler übermitteln.

Diese Überlegungen zeigen bereits den Aufwand der Adventureprogrammierung, glücklicherweise lassen sich aber auch zahlreiche unterschiedliche Spielereingaben von ein und derselben Programmzeile beantworten.

So wird unser Programmteil, der die untersuchenden Handlungen des Spielers bearbeitet, zunächst prüfen, ob der Gegenstand, über den nähere Informationen gewünscht werden, entweder in Reichweite des Spielers, somit im gleichen Raum, oder gar in dessen Besitz ist, ansonsten kann sofort mit der Eingabe des nächsten Zuges begonnen werden:

```
5000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 5900
5900 REM       GEGENSTAND NICHT VORHANDEN
5990 PRINT"So etwas sehe ich hier nicht
!":GOTO 1080
```

DIE BEDINGUNGEN

Abgesehen von diesen zwei Bedingungen, lassen sich eine Reihe weiterer Voraussetzungen formulieren die als Kriterium für die Ausführung einer Aktion herangezogen werden können, und ohne die ein Adventure nicht zu realisieren sein wird:

Objekt ist im Raum
Objekt wird vom Spieler mitgeführt
Objekt ist nicht im Raum
Flag ist gesetzt
Flag ist nicht gesetzt
Spieler ist in einem bestimmten Raum

Wie wir später noch sehen werden, erhebt diese Liste keinen Anspruch auf Vollständigkeit, denkbar ist beispielsweise eine Invertierung der letzten Bedingung, bestimmte Aktionen können in einem gewissen Raum nicht ausgeführt werden, wohl aber in jedem anderen Raum.

Einer Erläuterung bedürfen die an dieser Stelle aufgetauchten Flags. Es handelt sich hierbei um weitere Signalschalter, die hier jedoch unmittelbar der Kontrolle des Spielverlaufes dienen werden, und die wir mittels eines weiteren Feldes realisieren wollen. Auch diese Feldvariablen sollen nur zwei Zustände annehmen können, entweder ist ein Flag gesetzt, dann soll der Inhalt der Variablen -1 sein, oder es ist nicht gesetzt, was einer Null entsprechen soll.
Praktisch nutzen werden wir sie zur Identifizierung bestimmter Zustände:

ist eine Tür geöffnet oder verschlossen ?
wurde ein Hindernis beseitigt oder nicht ?
soll ein Ungeheuer dem Treiben des Spielers ein Ende setzen, oder darf er weiterziehen ?

In der Programmpraxis wird die Formulierung einer einzigen Bedingung meist jedoch nicht zur eindeutigen Klassifizierung der Situation ausreichen, so daß logische Verknüpfungen mehrerer Bedingungen erforderlich werden, um zu gewährleisten, daß einige Handlungen nicht völlig zusammenhanglos und zu einem falschen Zeitpunkt durchgeführt werden.
Bei diesen logischen Operationen wird es sich um UND- beziehungsweise ODER- Verknüpfungen handeln, für unsere Adventurepraxis heißt das, entweder müssen beide (AND) oder nur die eine oder die andere (OR) von zwei oder mehreren Verknüpfungen erfüllt sein. Können wir nicht auf mehr als zwei Bedingungen verzichten, wird meist auch eine Klammerung unumgänglich sein. Ein Beispiel soll dies deutlich machen:

Denken wir an den Sprengstoff in unserer Kiste. Er soll dazu benutzt werden, allzu voreiligen Spielern die Freude am schnellen Weiterkommen zu vergällen.
Eine Warnung ist vor der Explosion aus Fairneßgründen jedoch durchaus angebracht, weshalb auf *Untersuche Sprengstoff* die Meldung *Er sieht sehr explosiv aus* ausgegeben werden soll. Wenn der Spieler dann noch versucht, ihn zu nehmen, trägt er selbst die Schuld an seinem Ende.

Hat der Spieler tatsächlich obige Eingabe gemacht, wird der Treiber die Nummern 1 (untersuche) und 9 (Sprengstoff) zur Verfügung stellen. Unser Problem liegt nun darin, daß wir uns nicht mit der Überprüfung eines Raumes zufrieden geben können, denn wir dürfen nicht vergessen, daß der Spieler zwar den Sprengstoff nicht greifen und mitnehmen kann, wohl aber die Holzkiste, und wenn er die Kiste mit sich führt, hat er auch den Sprengstoff bei sich. Prüfen wir daher, ob der Spieler den Sprengstoff gemeint hat, ***UND***

gleichzeitig die Holzkiste (Objekt Nummer 7) im gerade besuchten Raum **ODER** im Inventory des Abenteurers ist. Der eigentliche Handlungsgegenstand (Nummer 9) steht nicht zur Verfügung, nach Zeile 5000 wird der Programmablauf daher mit Zeile 5900 fortgestetzt werden.

So muß die Ausführung des Befehles *Untersuche Sprengstoff* folgendermaßen formuliert werden:

```
5904 IF o=9 AND (ob(7)=spieler OR ob(7)=
-1) THEN PRINT"Er sieht sehr explosiv au
s !":GOTO 1080
```

DIE AKTIONEN

Das vorangegangene Beispiel hat uns die Ausführung der wohl einfachsten Aktion, die Ausgabe einer Mitteilung an den Spieler, deutlich gemacht.
Meist ist dieser Print - Befehl jedoch nur Beiwerk zu anderen Handlungen, wie Manipulationen von OB(), um dem Abenteurer in einer Rückmeldung die Ausführung des Befehles zu bestätigen.

Im Gegensatz dazu bedeuten Befehle wie Nimm, Leg, Zerstöre und auch Öffne für einen Gegenstand immer eine Ortsveränderung. Entweder wird er neuerdings vom Spieler mitgeführt, oder er gehört nun zum Inventar eines bestimmten Raumes, vielleicht verschwindet dieser Gegenstand auch ganz aus dem Spiel. Stellen Sie sich nur die Verblüffung des Spielers von Goldrausch vor, wenn der Bär den Honig offensichtlich genossen hat, die Flasche aber immer noch irgendwo zu finden ist !

Eine andere, ebenfalls häufig benötigte Aktion ist das Setzen oder Löschen der zuvor erwähnten Flags, um

stufenweise die Voraussetzungen für spätere Handlungen zu schaffen.

Vergessen dürfen wir auch auf keinen Fall bedingte Ortswechsel des Spielers, welche durch Zauberei oder harmlose Aktionen ausgelöst werden. Eine spätere Erweiterung von Goldrausch könnte aus der Höhle des Bären ein riesiges Labyrinth von Gängen machen, und nach *Untersuche Höhle* befindet sich der Spieler mitten in der Höhle, obwohl er diese nicht durch eine Bewegung in die entsprechende Himmelsrichtung betreten hat. Aber durch den erhöhten Programmieraufwand wird unser Adventure nur realistischer werden, denn wie soll man eine Höhle gründlich untersuchen, ohne sie zu betreten ?

Ähnlich wie bei den Bedingungen, können wir auch hier eine Liste möglicher Aktionen zusammenstellen, wobei allerdings wiederum kein Anspruch auf Vollständigkeit erhoben wird. Jedoch handelt es sich dabei um einen ausreichenden Grundstock, denn es ist möglich, allein mit den vorgestellten Bedingungen und Aktionen gute Adventures zu schreiben.

Mitteilung an den Spieler ausgeben
Objekt verschwindet
Objekt kommt ins Inventory
Objekt erscheint neu im Raum
Flag wird gesetzt
Flag wird gelöscht
Spieler wird in anderen Raum versetzt

PROGRAMMIERUNG DER BEFEHLSAUSFÜHRUNG

Im folgenden finden Sie Hinweise zu Besonderheiten, die bei einzelnen Befehlen beachtet werden müssen. Dabei werde ich nur einige typische Programmzeilen erläutern, entnehmen Sie die restlichen Zeilen bitte dem Listing der Miniversion von Goldrausch am Ende dieses Kapitels.

Aktion: Untersuche

Zunächst steht fest, daß das Ergebnis der Untersuchung eines Gegenstandes immer in Form eines Satzes mitgeteilt wird.
Häufig wird es sich dabei um ein und denselben Satz handeln (Ich sehe nichts besonderes), weshalb wir die Mitteilungen an den Spieler in solchen Fällen nicht direkt ausgeben, sondern wir weisen den Text bei Programmstart einer Variablen zu und drucken deren Inhalt zu gegebener Zeit aus.

```
600 REM ******************** MITTEILUNGEN
601 m$(1)="Ich sehe nichts besonderes."
602 m$(2)="So stark bin ich nicht !"
603 m$(3)="Wie stellst Du Dir das vor ?"
604 m$(4)="Der Baer greift sich den Honi
g, und"
605 m$(5)="verschwindet in der Tiefe der
 Hoehle."

690 ok$="O.k."
```

Bereits erläutert wurde Zeile 5000, sie stellt für den gerade aktuellen Gegenstand durch Überprüfung des Wertes in

OB() sicher, daß das Objekt sich in der Nähe des Spielers befindet.
Trifft dies zu, wird entsprechend der Objektnummer o die für diesen Gegenstand zuständige Programmzeile ermittelt. Sind alle dort programmierten Bedingungen erfüllt, wird der Rest der Zeile abgearbeitet und der Programmlauf anschließend mit dem Neuaufbau des Bildschirmbildes fortgesetzt:

```
5002 IF o=1 THEN PRINT m$(1):GOTO 1080
5003 IF o=3 THEN PRINT m$(1):GOTO 1080
5004 IF o=4 THEN PRINT"In einer Ecke ste
ht ein Regal.":OB(8)=spieler:GOTO 1080
```

Wird die Holzhütte untersucht (o=4), entdeckt der Spieler das Regal. Dieses war bislang in Raum 0, dem Lagerraum, und wird nun im Raum des Spielers positioniert, weshalb der Adventuretreiber beim anschließenden Durchlaufen der Zeilen 1160 bis 1210 unter anderem auch *ein klappriges Regal* ausgibt.

Nun ist es aber, wie im Fall des Regales, denkbar, daß ein Objekt auftaucht, welches der Spieler anschließend an sich nimmt.
Dann wäre eine Formulierung wie in 5004 nicht mehr ausreichend, denn die Flasche würde bei jeder weiteren Untersuchung des Regales dem Inventory automatisch entnommen und wieder in den Raum gestellt werden.

Diesen Fehler können wir nur vermeiden, wenn wir uns zuvor vergewissern, daß die Flasche noch nicht im Spiel war, somit also noch immer im Lagerraum ist.

```
5008 IF o=8 AND ob(5)=0 THEN PRINT"Auf d
em Regal steht eine Korbflasche.":ob(5)=
spieler:GOTO 1080
5009 IF o=8 AND ob(5)<>0 THEN PRINT m$(1
):GOTO 1080
```

Bei allen weiteren Untersuchungen des Regales wird der Spieler nichts besonderes mehr feststellen (Zeile 5009).

Ein weiteres Beispiel soll den Gebrauch der Flags zeigen. Hat der Spieler die Schatztruhe gefunden, ist diese zunächst mittels einer Eisenkette verschlossen. Nachdem er dieses Hindernis beseitigt hat, muß die Truhe selbstverständlich erst geöffnet werden, ehe der wertvolle Inhalt sichtbar wird. Um diese drei Zustände dem Spieler deutlich zu machen, wäre es möglich, weitere Gegenstände, wie eine mit einer Eisenkette verschlossene Truhe, eine unverschlossene Truhe, und eine geöffnete Truhe, einzuführen. Dieser recht hohe Programmieraufwand, der nebenbei natürlich wiederum den zur Verfügung stehenden Speicherplatz einschränkt, läßt sich durch den Einsatz von Signalschaltern umgehen. Sinnvollerweise fertigen Sie während der Konstruktion Ihres Adventures eine Tabelle an, damit Sie auch wirklich die richtigen Flags umstellen:

Flag	0	-1
1	Kette heil	Kette zerstört
2	Truhe zu	Truhe geöffnet

Warum als Inhalt nur -1 und 0, und warum nicht nur ein einziges Flag mit verschiedenen Ziffern für jeden möglichen Zustand vorgesehen worden ist, können Sie sich inzwischen selbst erklären.
Denn wir machen uns die Tatsache zunutze, daß für den Basic - Interpreter ein wahrer Ausdruck immer durch eine -1 repräsentiert wird, und können somit unsere Programmzeilen wieder etwas kürzer halten, denn in den IF - Statements müssen wir keine Vergleichswerte angeben:

```
IF FL(1)=-1   entspricht  IF FL(1)
IF FL(1)= 0   entspricht  IF NOT FL(1)
```

Für unser Beispiel gelten demnach folgende Zeilen:

```
5011 IF o=11  AND NOT fl(1) THEN PRINT"S
ie ist mit einer Eisenkette ver-":PRINT"
SCHLOSSEN.":GOTO 1080
5012 IF o=11 AND fl(1) AND NOT fl(2) THE
N PRINT"Aussen sehe ich nichts besondere
s.":GOTO 1080
5013 IF o=11 AND fl(1) AND fl(2) AND ob(
12)=0 THEN PRINT"Sie ist voller Silbermu
enzen.":ob(12)=spieler:GOTO 1080
```

Ein weiterer Punkt, dem wir besondere Beachtung schenken müssen, findet seinen Grund in der Auslegung unserer Wortanalyse.
Zur Ausstattung der ersten zwei Räume gehören die völlig identischen Objekte 1 und 2. Wird die Eingabe des Spieles überprüft, so wird als Objektnummer immer nur eine 1 ermittelt werden können, welche bei einem Aufenthalt des Spielers in Raum 2 nicht den Tatsachen entspricht.
Um Eingabefehler richtig quittieren zu können, hatten wir jedoch Programmzeile 5000 eingesetzt, welche uns nun auch die Behandlung dieser Sonderfälle ermöglicht.
So lösen wir das Problem *Untersuche Baeume* für Raum 2 folgendermaßen:

```
5901 IF o=1 AND spieler=2 THEN PRINT m$(
1):GOTO 1080
```

Zum Abschluß der Routine müssen wir eine letzte Zeile vorsehen, die, wenn in keiner der vorangegangenen Programmzeilen alle Bedingungen erfüllt waren, ohne jede Bedingung durchgeführt wird. Hiermit werden nicht definierte Spieleingaben abgefangen, und wir verhindern, daß, wenn keine zutreffende Bedingung gefunden wurde, die Zeilen der anderen Verben abgearbeitet werden und somit der Programmlauf außer Kontrolle gerät. Selbst wenn wir eine

Handlung, die für den Ablauf des Adventures nicht notwendig ist, vergessen, wird ein Spieler das niemals bemerken:

```
5990 PRINT"So etwas sehe ich hier nicht
!":GOTO 1080
```

Aktion: NIMM

Genau so, wie der Spieler einen Gegenstand nur untersuchen kann, der sich in seiner Nähe befindet, so wird er ihn auch nur nehmen können, wenn der Gegenstand entweder im gleichen Raum wie der Spieler oder bereits in dessen Taschen ist.

```
6000 IF ob(o)<>spieler AND ob(n)<>-1 THE
N GOTO 6900
```

Danach wäre es an sich ganz einfach, den Befehl auszuführen, für eine Reihe von Objekten sollten wir dem Abentuerer seinen Wunsch jedoch versagen.
So wollen wir berücksichtigen, daß eine vollgefüllte Eisentruhe vermutlich zu gewichtig für einen einzelnen Menschen ist, ebenso können wir selbst dumme Versuche des Spielers wie *Nimm Baum* nicht unbeachtet lassen.

```
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6005 IF o=11 THEN PRINT m$(2):GOTO 1080
```

Natürlich kann es auch noch schlimmer kommen, denn wir müssen dem Spieler selbstverständlich eine ausreichende Anzahl von Fallen stellen, damit ihm der Sieg nicht zu einfach gemacht wird. So sollte laut unserem Drehbuch eine Berührung des Sprengstoffes zur Explosion und damit zum Spielende führen. Gleiches gilt für den Fall, daß er versucht, sich der Nuggets zu bemächtigen, ohne dem Bären zuvor einen Leckerbissen angeboten zu haben. Ebenso fatal

für den Spieler müßte sein Versuch sein, den Bären zu nehmen:

```
6015 IF o=1 AND spieler=5 THEN m$(0)="De
r Baer hat mich erschlagen.":GOTO 4500
6018 IF o=17 AND NOT fl(3) THEN m$(0)="E
in Baer stuerzt sich auf mich.":GOTO 450
0
6900 IF o=9 AND (ob(7)=spieler OR ob(7)=
-1) THEN m$(0)="Bei der Beruehrung ist d
er Sprengstoff explodiert !":GOTO 4500
```

Anmerkung: In Zeile 4500 beginnt eine Routine, welche bei nicht gewonnenem Spiel aufgerufen wird. m$(0) ist in dieser Routine für eine erläuternde Nachricht vorgesehen.

Flag 3 wird gesetzt, sobald der Bär in den Besitz des Honigs gelangt.

Steht dem Versuch des Spielers, sich an irgendwelchen Gegenständen unseres Adventures zu bereichern, nichts im Wege, müssen wir dem betreffenden Objekt nur die neue Platznummer, eine -1, zuweisen:

```
6010 IF o=5 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
```

Aktion: LEG

Irgendwann jedoch wird der Spieler sich dieser Objekte auch wieder entledigen wollen, entweder, weil wir als Programmierer seine Tragfähigkeit begrenzt haben, oder weil er sie irgendwie benutzen will.
In der Praxis wirft die Leg weg - Routine kaum Probleme auf, darüberhinaus überzeugt sie durch ihre Kürze.
Denn die einzige Voraussetzung ist, daß der wegzulegende Gegenstand im Besitz des Spielers ist, was für alle Objekte in Zeile 7000 getestet wird:

```
7000 IF ob(o)<>-1 THEN PRINT"So etwas ha
be ich doch gar nicht !":GOTO 1080
7900 ob(o)=spieler:PRINT ok$:GOTO 1080
```

Irrtümer des Spielenden werden sofort erkannt und entsprechend quittiert. War der Gegenstand im Inventory, wird ihm mit einer Änderung des Inhaltes von OB() ein neuer Aufenthaltsort zugewiesen.

Prinzipiell reichen diese beiden Zeilen aus, es kann jedoch erforderlich sein, neben dem Positionswechsel weitere Aktionen durchzuführen. So hatten wir für Goldrausch vorgesehen, daß der Bär sich die Flasche greift und damit in den Tiefen der Höhle verschwindet.
Diese Aktion darf natürlich nur durchgeführt werden, wenn der Spieler sich gerade in Raum 5 befindet, an allen anderen Orten wird der Vorgang ebenfalls durch Zeile 7900 ausgeführt.

```
7020 IF o=5 AND spieler=5 THEN ob(5)=0:f
l(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0
:GOTO 1080
```

Aktion: OEFFNE

Gleich zu Beginn dieses Programmteiles steht der nun schon bekannte Test, um technische Fehler zu vermeiden. Wiederum ist es nötig, sowohl den Raum als auch das Inventory zu überprüfen, schließlich wird der Spieler eine zu öffnende Tür nicht erst nehmen müssen, was aber für eine Flasche durchaus Bedingung sein kann:

```
8000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N PRINT"So etwas ist hier nicht.":GOTO 1
080
```

Die Aktion selbst wird meist aus dem Setzen eines Flags und der Ausgabe einer Mitteilung bestehen:

```
8025 IF o=11 AND fl(1) THEN PRINT ok$;"
- der Deckel klappt nach hinten.":fl(2)=
-1:GOTO 1080
```

Nicht vergessen dürfen wir unwichtige Aktionen, die aber von der Lage der Dinge her möglich sind. Als Beispiel mag noch einmal unsere Flasche dienen. Ein logisch denkender Abenteurer wird sie vor einer näheren Untersuchung erst öffnen wollen und wäre sicher überrascht, wenn ihm dies nicht gelingt bloß weil wir es für unrelevant in Bezug auf den weiteren Spielverlauf gehalten haben:

```
8010 IF o=5 THEN PRINT ok$:GOTO 1080
```

Der Abschluß dieses Blockes gewährleistet, daß neben den technischen Falscheingaben auch inhaltliche Unmöglichkeiten wie *Öffne Honig* erkannt werden.

```
8999 PRINT"Ich verstehe nicht, was Du me
inst.":GOTO 1080
```

ZUSAMMENFASSUNG

Ich hoffe, daß Ihnen mit den Erläuterungen zu diesen wohl häufigsten Handlungen die Ausführung der eigentlichen Spielprogrammierung klar geworden ist. Prinzipiell läßt sich zusammenfassend sagen, daß die einzelnen Aktionen durch zwei Programmzeilen umklammert werden.
So wird zu Beginn eines Blockes getestet, ob von der technischen Seite her die Durchführung der Aktion möglich wird, abschließend wird sichergestellt, das der Programmlauf selbst bei Auftreten irgendeines Fehlers korrekt fortgesetzt wird.
Das Erkennen der richtigen Voraussetzungen für die Durchführung eines Befehles ist dabei, ebenso wie die Aktion selbst, meist eine recht einfach zu programmierende Angelegenheit, die bei unserem Adventuresystem zudem noch standardisiert wurde. Folgende Zusammenstellung soll Ihnen bei der Realisation Ihrer ersten eigenen Adventures eine Hilfe sein:

BEDINGUNG	***IM BASICPROGRAMM***
Objekt ist im Raum	ob(objekt)=sp
Objekt wird vom Spieler mitgeführt	ob(objekt)=-1
Objekt ist nicht im Raum	ob(objekt)<>sp
Flag ist gesetzt	fl(x)=-1
Flag ist nicht gesetzt	fl(x)=0
Spieler ist in einem bestimmten Raum	sp=Raumnummer

AKTIONEN	***IM BASICPROGRAMM***
Mitteilung an den Spieler ausgeben	PRINT" oder M$
Objekt verschwindet	ob(Objekt)=0
Objekt kommt ins Inventory	ob(Objekt)=-1
Objekt erscheint neu im Raum	ob(Objekt)=sp
Flag wird gesetzt	fl(x)=-1
Flag wird gelöscht	fl(x)=0
Spieler wird in anderen Raum versetzt	sp=Raumnummer

DIE LETZTEN SCHRITTE

Bevor unser Adventure den in Kapitel 2 entwickelten Vorstellungen entspricht, müssen wir uns noch Gedanken zu drei weiteren Programmteilen machen.
So ist es für einen einwandfreien Spielablauf unbedingt erforderlich, dem Spieler die Möglichkeit zu einer schnellen Inventur in die Hand zu geben.

Es wäre duchaus möglich, diese Aktion auf gleiche Art und Weise wie die Behandlung der übrigen Verben durchzuführen. Da es sich hierbei jedoch um eine grundsätzliche Funktion der Abenteuerprogramme handelt, werden wir unseren Treiber in geeigneter Weise ergänzen.

Inventur

Inventur gehört, wie auch die später zu erstellenden Routinen zum Abspeichern und Laden des Spielstandes, zur Befehlsgruppe der sogenannten Ein - Wort - Befehle. Für diesen Programmteil haben wir zwischen den Programmzeilen 1480 (Ende der Bewegungsroutine) und 2000 (Analyse der Eingabe) ausreichend Platz gelassen.

Um unseren Treiber nicht unnötig lange damit zu beschäftigen, einen jeden Spielerzug mit den Befehlen dieser Sondergruppe zu vergleichen, entscheiden wir zunächst, ob es sich um eine dieser speziellen Anweisungen oder um eine reguläre Eingabe handelt.
Die übliche Spielereingabe wird eine Länge von mindestens neun Buchstaben haben, gehen wir daher davon aus, daß es sich, wenn die Länge der Eingabe geringer ist, um einen Ein - Wort Befehl handelt:

```
1480 IF LEN(eingabe$)>8 THEN GOTO 2000
```

Handelte es sich bei der Eingabe des Spielers nicht um eine reguläre Verb/Objekt - Kombination, dann sollen alle Zeilen, in denen die Behandlung eines EW - Befehles beginnt, der Reihe nach durchlaufen werden, bis schließlich eine Übereinstimmung der ersten drei Buchstaben festgestellt wird.

In der Praxis besteht die Ausführung der Inventur aus einer einfachen Schleife, welche alle Gegenstände des Adventures, deren Aufenthaltsort mit -1 gekennzeichnet ist, auf dem Bildschirm ausdruckt:

```
1499 REM **************** START INVENTUR
1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 2000
1510 PRINT"Ich trage folgendes mit mir:"
1520 FOR objekt=1 TO ao
1530 IF ob(objekt)=-1 THEN PRINT ob$(obj
ekt)
1540 NEXT objekt
1550 GOTO 1080
1560 REM ***************** ENDE INVENTUR
```

Die Titel

Rein technisch gesehen haben wir unser Programm mit diesen Zeilen fertiggestellt. Alle gewünschten Funktionen stehen zur Verfügung und einem Spiel steht nichts mehr im Wege. Doch wie lange soll der Abenteurer in unserer Welt umherirren, wann hat er sein Ziel erreicht ?

Handlungsmäßig ist die Miniversion von Goldrausch beendet, wenn sowohl die Nuggets als auch die Silberstücke sich im Besitz des Spielers befinden.

Eine einzige Basic - Zeile reicht aus, um in diesem Falle alle weiteren Eingaben zu unterbinden und einen Programmteil anzuspringen, welcher dem Spieler in angemessener Form seinen Erfolg mitteilt und das Spiel beendet.

```
1340 IF ob(12)=-1 AND ob(17)=-1 THEN GOT
O 4800
```

Für Adventurespiele typisch ist jedoch ein völlig anderes, weniger ruhmreiches Ende. Da auch Spieler unserer Programme meistens gezwungenermaßen ihr Spiel etwas früher beenden werden, programmieren wir ein weiteres Endbild ab Zeile 4500.
Bei der Ausführung der spielbeendenden Handlungen hatten wir ja bereits eine entsprechende Nachricht vorbereitet, so daß der Spieler zumindest nicht über die Ursache seines Scheiterns im Unklaren gelassen werden muß.

```
4500 CLS:REM                    SPIELER TOD
4600 PRINT"Auch das noch !":PRINT:PRINT
m$(0)
4610 PRINT:PRINT"Ich bin tod !":PRINT
4620 INPUT"Soll ich es noch einmal versu
chen ";eingabe$:eingabe$=UPPER$(eingabe$
)4630 IF LEFT$(eingabe$,1)="J" THEN RUN
4640 GOTO 1960
```

Dabei erlaubt es uns die Einbeziehung der individuellen Nachricht in m$(0), dieses Titelbild für jede Situation und für jedes Adventure zu verwenden. Änderungen werden, sofern sie überhaupt erforderlich sind, nur wenig Aufwand verursachen. Aus diesem Grunde können obige Zeilen, ebenso wie die Siegesnachricht ab 4800, als weitere Ergänzungen unseres Adventuretreibers betrachtet werden.

Bedenken wir weiterhin, daß es für Adventures durchaus nicht typisch ist, sie zu laden und in einigen wenigen

Stunden durchzuspielen, und programmieren wir daher zu guter Letzt noch ein reguläres Programmende.

Auch wenn sich solche Software auf dem Markt befindet, es zeugt nicht von der Benutzerfreundlichkeit eines Programmes, wenn es nur durch Drücken der Break- oder Resettaste verlassen werden kann.

Fügen wir daher einen weiteren Ein - Wort Befehl in unser System ein:

```
1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1950
1950 IF LEFT$(eingabe$,3)<>"END" THEN GO
TO 2000 ELSE CLS
1960 PRINT "Der Autor wuenscht Ihnen meh
r Glueck":PRINT"beim naechsten Mal !":PR
INT:PRINT:PRINT:END
```

Sinngemäß waren das die letzten Zeilen unseres Adventures, alle denkbaren Versionen des Spielendes haben wir realisiert. Doch wie nimmt sich dagegen der Beginn unseres Werkes aus ?

Starten wir unser Programm so wie es ist, besteht dessen erste Handlung darin, die Variablen vorzubereiten, eine Arbeit, die besonders bei größeren Adventures den Gedanken an einen Absturz des Programmes aufkommen läßt. Anschließend befindet man sich mitten im Programm, und ein unerfahrener Adventureneuling sieht dann möglicherweise den Sinn des Spieles darin, eine Seight Seeing Tour durch eine elektronisch simulierte Welt anzustellen.

Schenken wir daher einer alten Volksweisheit Glauben, die besagt, daß der erste Eindruck oft entscheidend ist.
Machen wir uns die Mühe, unser Programm, bevor wir es als fertig bezeichnen, mit einem, oder besser noch, mit mehreren Titeln zu versehen.

Dem ersten Titel wird dabei die Aufgabe zufallen, in knapper, aber ansprechender Form, Aufklärung über die im Speicher befindlichen Bytes zu geben, Programmtitel und -gattung zu nennen sowie Auskunft über den Autoren zu geben.

Das nächste Bild bringen wir auf den Schirm, bevor die Variablen initialisiert werden. Während dieses Zeitraumes kann dieser Titel Auskunft über die dem Spieler gestellte Aufgabe geben, und ihn in die Handlung einführen.

```
          Herzlich Willkommen zu
               "GOLDRAUSCH"
----------------------------------------
Auf der Suche nach dem Glueck in der
neuen Welt begegneten Sie vor einigen
Tagen einem alten todkranken Mann, dem
Sie in seinen letzten Stunden Beistand
leisteten.
Aus Dankbarkeit brichtete er Ihnen von
seiner Goldmine und den dort versteckten
Resten seines Vermoegens.
Zahllosen Gefahren widerstanden Sie auf
dem Weg dorthin; bald werden Sie Ihr
Ziel erreicht haben und es wird sich
zeigen, ob der Alte die Wahrheit
gesprochen, oder im Fiebertraum geredet
hatte.
----------------------------------------
   Wuenschen Sie Ratschlaege fuer Ihr
           weiteres Vorgehen ?
```

Gehört das Herausfinden der gestellten Aufgabe allerdings mit zur Spielidee, wird diese Tafel fehlen, und an den Programmtitel schließt sich gleich eine Erklärung der Adventurespiele an, die ansonsten als drittes Titelbild implementiert werden muß.

Denn wie war es, als Sie Ihr erstes Adventure in die Hand und in den Speicher bekamen, kannten Sie Sinn und Ziel dieser Programme, oder saßen Sie rat- und tatenlos vor Ihrem Computer ?

Vergessen wir doch nicht, daß es immer wieder Anfänger geben wird. Anfänger, die froh sind, wenn sie nach dem Einschalten des Rechners ein Programm ohne Fehlermeldungen laden und starten können. Eine grundsätzliche Spielanweisung bedeutet doch nur geringen zusätzlichen Aufwand, machen wir uns daher ans Werk und rechtfertigen ein mögliches Fehlen nicht etwa damit, daß das Herausfinden der Spielregeln bereits das erste beabsichtigte Hindernis wäre.

```
CPC - Ventures
          (c) 1984  by Jörg Walkowiak
-----------------------------------------
Stellen Sie sich einen Roboter  vor, den
Sie  mit zahlreichen  Kommandos  steuern
koennen.
Ich bin  dieser  Roboter, und  ich werde
mich fuer Sie den Gefahren der verwegen-
sten Abenteuer aussetzen.
Damit Sie  mich sinnvoll  agieren lassen
koennen, werde ich  Ihnen  die Situation
in der ich mich gerade  befinde, jeweils
genau beschreiben.
Anschliessend  sagen  Sie  mir  mit zwei
Worten,  wie  beispielsweise  UNTERSUCHE
TUER, NIMM MESSER, was ich tun soll.

Darueber hinaus verstehe ich die Befehle
     INVENTUR, SCORE, VOKABELN, HELP
         SAVE & LOAD sowie ENDE.
```

Zwei Beispiele haben Ihnen gezeigt, wie es gemeint war. Für diese Aufgaben stehen die Programmzeilen 10 - 100, sowie 700 - 900 Verfügung.
Sinnvollerweise werden die allgemeinen Erläuterungen dabei als Unterprogramm eingebaut, was den weiteren Ausbau des Adventuretreibers um den Befehl INStruktionen erlaubt. Der Abenteurer kann sich nun jederzeit ohne einen Abbruch des Spieles die allgemeinen Regeln in die Erinnerung zurückrufen.

HINWEISE ZUM FOLGENDEN LISTING

Nachdem Sie die folgenden Zeilen in Ihren Schneider CPC eingegeben haben, steht Ihnen die lauffähige Miniversion von Goldrausch zur Verfügung. Es gilt jedoch zu beachten, daß es sich nur um zusätzliche Programmzeilen handelt, alle im vorangegangenen Text erläuterten Zeilen müssen sich bereits im Speicher befinden.

Sie werden sehen, daß mit den bisher vorgestellten Techniken bereits anspruchsvolle Adventures entwickelt werden können, die den Vergleich mit entsprechender käuflicher Software nicht zu scheuen brauchen.

Es liegt dann an Ihnen, ob Sie das Buch zunächst beiseite legen und ein eigenes Abenteuerspiel entwickeln oder lieber weitere Features kennenlernen wollen, die aus einem guten ein perfektes Adventure machen können.

```
20 CLS:PLOT 420, 170:DRAW 420, 360:DRAW
640, 360:DRAW 640, 340:DRAW 450, 340:DRA
W 450, 170:DRAW 460, 180:DRAW 460, 340:P
LOT 460, 200:DRAW 530, 270:PLOT 530, 340
:DRAW 530, 260:DRAW 540, 260:DRAW 540, 3
40:PLOT 540, 260
21 DRAW 550, 270:DRAW 550, 340:PLOT 550,
 290:DRAW 600, 340:PLOT 640, 340:DRAW 30
0, 0:PLOT 440, 0:DRAW 640, 200:PLOT 640,
 200:PLOT 500, 200:DRAW 640, 200:PLOT 60
0, 160:DRAW 460, 160:PLOT 420, 120:DRAW
560, 120
22 PLOT 520, 80:DRAW 380, 80:PLOT 340, 4
0:DRAW 480, 40:PLOT 440, 0:DRAW 300, 0:P
LOT 540, 240:DRAW 640, 240:PLOT 580, 280
:DRAW 640, 280:PLOT 620, 320:DRAW 640, 3
20:PLOT 450, 170:DRAW 420, 170:PLOT 420,
 360
23 DRAW 440, 380:DRAW 640, 380:PLOT 430,
 370:DRAW 430, 380:DRAW 440, 390:DRAW 44
0, 400:PLOT 280, 150:DRAW 150, 150:DRAW
150, 200:DRAW 280, 200:DRAW 280, 150:DRA
W 310, 180:PLOT 310, 180:PLOT 280, 200:D
RAW 310, 230
24 DRAW 310, 180:PLOT 310, 230:DRAW 180,
 230:DRAW 150, 200:PLOT 150, 200:DRAW 16
0, 210:DRAW 270, 210:DRAW 280, 200:PLOT
270, 210:DRAW 290, 230:PLOT 270, 190:DRA
W 160, 190:DRAW 160, 160:DRAW 270, 160:D
RAW 270, 190
25 PRINT CHR$(150);STRING$(18,154);CHR$(
156)
26 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
27 PRINT CHR$(149);"    GOLDRAUSCH    ";
CHR$(149)
28 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
29 PRINT CHR$(149);"ein Adventure  von";
CHR$(149)
30 PRINT CHR$(149);"  J";CHR$(178);"rg W
```

```
alkowiak  ";CHR$(149)
31 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
32 PRINT CHR$(147);STRING$(18,154);CHR$(
153)
33 LOCATE 3,25:PRINT CHR$(164);" 1984  b
y DATA BECKER, Duesseldorf";
40 FOR i=1 TO 8000:NEXT
120 ao=19
140 af=3
160 wortlaenge=4
318 DATA"eine Eisenstange","EISE",0
319 DATA"eine Eisenkette","KETT",0
699 REM *********** 2. TITEL: EINLEITUNG
700 CLS:PEN 1:PRINT"Herzlich Willkommen
zur Miniversion von":PRINT:PEN 2:PRINT T
AB(15)CHR$(34);"GOLDRAUSCH";CHR$(34):PRI
NT:PEN 1
705 PRINT STRING$(40,208)
710 PRINT"Auf  der Suche  nach  dem Glue
ck in  derneuen  Welt  begegneten Sie  v
or einigenTagen einem  alten  todkranken
 Mann, demSie in  seinen letzten  Stunde
n Beistandleisteten."
720 PRINT"Aus Dankbarkeit  berichtete er
 Ihnen vonseiner Goldmine und den dort v
ersteckten Resten seines Vermoegens."
730 PRINT"Zahllosen Gefahren  widerstand
en Sie aufdem  Weg  dorthin;  bald  werd
en Sie IhrZiel erreicht  haben  und  es
 wird sichzeigen,  ob  der  Alte  die Wa
hrheit ge-sprochen,  oder  im  Fiebertra
um geredethatte."
740 PRINT STRING$(40,210)
800 REM ************ SPIELDATEN EINLESEN
880 PRINT:INPUT"   Wuenschen Sie Ratschl
aege fuer Ihr               weiteres Vorg
ehen";eingabe$:eingabe$=UPPER$(eingabe$)
890 IF LEFT$(eingabe$,1)="J" THEN GOSUB
900
895 GOTO 1000
```

```
899 REM ******** 3. TITEL: INSTRUKTIONEN
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
1:INK 1,25
910 PRINT"        CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J"
;CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Stellen
Sie  sich einen Roboter vor, denSie  mit
 zahlreichen  Kommandos  steuernkoennen.
"
930 PRINT"Ich bin dieser  Roboter,  und
ich  werdemich fuer Sie den Gefahren der
 verwegen-sten Abenteuer aussetzen."
940 PRINT"Damit Sie  mich sinnvoll  agie
ren lassenkoennen,  werde ich  Ihnen die
 Situationin der ich  mich gerade befind
e, jeweilsgenau beschreiben."
950 PRINT"Anschliessend  sagen  Sie  mir
  mit zweiWorten,  wie  beispielsweise
UNTERSUCHETUER, NIMM MESSER, was ich tun
 soll.":PRINT
960 PRINT"Darueber hinaus verstehe ich d
ie Befehle     INVENTUR, INSTRUKTIONEN u
nd ENDE.
970 PRINT STRING$(40,210)
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT
"              <TASTE>";:PEN 2:PRINT" drueck
en";:PEN 1:PRINT" ..."
990 eingabe$=INKEY$:IF eingabe$="" THEN
990 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK
 3,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1900
1900 IF LEFT$(eingabe$,3)<>"INS" THEN GO
TO 1950
1910 GOSUB 900
1920 GOTO 1080
2190 REM **** SZPRUNGZIEEE  unte nimm leg
```

```
oeffne benutze zerstoere
4800 CLS:REM                    SPIELER SIEGT
4810 PRINT"Herzlichen Glueckwunsch !"
4820 PRINT:PRINT"Sie haben die Ihnen ges
tellte Aufgabe":PRINT:PRINT"geloest und
duerfen sich nun an einem":PRINT:PRINT"a
nderen Adventure versuchen."
4830 PRINT:PRINT:PRINT:PRINT:PRINT:END
4999 REM *********** SPIELZUG AUSFUEHREN
5005 IF o=5 THEN PRINT"Die Flasche ist g
efuellt mit Honig.":GOTO 1080
5006 IF o=6 THEN PRINT m$(1):GOTO 1080
5007 IF o=7 AND ob(9)=0 THEN PRINT"In de
r Kiste liegt Sprengstoff.":GOTO 1080
5010 IF o=10 THEN PRINT"In dem Erdloch l
iegt eine Eisentruhe.":ob(11)=spieler:GO
TO 1080
5014 IF o=12 THEN PRINT"Genau das suche
ich !":GOTO 1080.
5015 IF o=13 THEN PRINT"Ich habe einen B
aeren aufgeschreckt !":ob(14)=spieler:fl
(3)=0:GOTO 1080
5017 IF o=15 THEN PRINT"Zwischen den Bue
schen ist ein Erdloch.":ob(10)=spieler:G
OTO 1080
5018 IF o=16 THEN PRINT"Sie sehen sehr s
tabil aus.":GOTO 1080
5019 IF o=17 THEN PRINT"Es handelt sich
um pures Gold !":GOTO 1080
5800 PRINT m$(1):GOTO 1080
5902 IF o=6 AND ob(5)=-1 THEN PRINT"Er i
st suess und gut.":GOTO 1080
5903 IF o=6 AND ob(5)<>-1 THEN PRINT"Ich
 habe keinen Honig !":GOTO 1080
5905 IF o=16 AND ob(18)=-1 THEN PRINT"Ka
um angerostet - wirklich stabil !":GOTO
1080
6002 IF o=3 THEN PRINT m$(2):GOTO 1080
6003 IF o=4 THEN PRINT m$(3):GOTO 1080
6004 IF o=8 THEN PRINT m$(2):GOTO 1080
6006 IF o=10 THEN PRINT m$(3):GOTO 1080
```

```
6007 IF o=13 THEN PRINT m$(3):GOTO 1080
6008 IF o=15 THEN PRINT m$(2):GOTO 1080
6011 IF o=6 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6014 IF o=12 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6017 IF o=17 AND fl(3) THEN PRINT ok$:ob
(o)=-1:GOTO 1080
6901 IF o=16 AND ob(18)=spieler THEN PRI
NT ok$:ob(18)=-1:GOTO 1080
6910 IF o=1 AND spieler=2 THEN PRINT m$(
2):GOTO 1080
6999 PRINT"So etwas sehe ich hier nicht
!":GOTO 1080
7000 IF o=16 AND ob(18)=-1 THEN PRINT ok
$:ob(18)=spieler:GOTO 1080
7001 IF ob(o)<>-1 THEN PRINT"So etwas be
sitze ich doch gar nicht !":GOTO 1080
7010 IF o=6 AND spieler=5 THEN ob(6)=0:f
l(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0
:GOTO 1080
8005 IF o=4 AND spieler=3 THEN PRINT"Die
 Huette war bereits offen.":GOTO 1080
8020 IF o=11 AND NOT fl(1) THEN PRINT"Da
s laesst die Kette nicht zu.":GOTO 1080
9000 IF o=16 AND ob(18)=-1 AND spieler=4
 THEN PRINT"Die Kette zerspringt.":fl(1)
=-1:GOTO 1080
9999 PRINT"Ich verstehe nicht, was Du me
inst.":GOTO 1080
10000 IF o=16 AND spieler=4 AND ob(18)=-
1 THEN PRINT"Die Kette zerspringt.":fl(1
)=-1:GOTO 1080
10010 IF o=16 AND spieler=4 AND ob(18)<>
-1 THEN PRINT"Womit ?":GOTO 1080
10999 PRINT"Ich verstehe nicht, was Du m
einst.":GOTO 1080
```

4. KAPITEL
- PERFEKTIONIERUNG -

VOM GUTEN ZUM PERFEKTEN ADVENTURE

Wenn Sie mit der Miniversion von Goldrausch gespielt oder selber ein Adventure auf der Basis der bislang erarbeiteten Techniken erstellt haben, werden Sie mit mir übereinstimmen, daß der Standard gängiger Programme der Gattung Adventures durchaus erreicht wurde.

Dennoch lassen sich ab und an Abenteuerspiele finden, die mit weiteren Extras aufwarten. Um sich über die Konkurrenz erheben zu können, wurde entweder ihre Benutzerfreundlichkeit erhöht, oder es fanden bei der Realisierung des Spieles auch die kleinen Nebensächlichkeiten des täglichen Lebens Verwendung, und erschweren dem Spieler das Leben nun zusätzlich.
Darüber hinaus existieren bereits einige Programme, die mit entsprechendem Werbeaufwand auf ihre noch nie dagewesenen Besonderheiten aufmerksam machen.
So wird auch dem Gehör des Spielers etwas geboten, von einzelnen Beeps und Pieps, die einen verstandenen oder aber einen undurchführbaren Befehl signalisieren über die akustische Untermalung sichtbarer Objekte bis hin zur Nachahmung der menschlichen Stimme. Leider ist der spielerische Nutzen all dieser Extras gleich Null, und man sollte es sich mehr als zweimal überlegen, ob man sein Geld für speicherfüllende Effekthascherei opfert oder lieber eine um den gesparten Speicherplatz vergrößerte Adventurewelt vorzieht.

BENUTZERFREUNDLICHKEIT

wird für Geschäftsprogramme immer wieder gefordert, warum nicht auch für Spiele ?

Da gibt es doch tatsächlich Adventures, die dem Spieler nicht einmal mitteilen, auf welchen Wegen er einen gefährlichen Ort wieder verlassen kann. Sie warten nur darauf, ihn durch die nach Eingabe der verschiedensten Himmelsrichtungen auftauchende Mitteilung ´In diese Richtung kannst du nicht. - You can´t go in that direction.´ zur Verzweiflung zu treiben.

Gut, daß der Spieler sich die Haare raufen darf und soll, damit bin ich einverstanden, doch scheinen mir diese Programme eher dazu geschrieben worden zu sein, um mit dem Werbespruch ´Wir garantieren Ihnen, daß Sie Monate brauchen, um alle Orte dieses Adventures kennenzulernen !´ an den Adventurefan gebracht zu werden.

Diese Kritik kann unser Adventuresystem glücklicherweise nicht treffen, dennoch ist mit dem Fehlen jeder Möglichkeit zum Abspeichern des augenblicklichen Spielstandes eine Voraussetzung gegeben, die Freude an einem Spiel rasch zu vergällen.
Unglücklicherweise gilt das umso mehr für Adventures, bei deren Entwicklung wir einen besonderen Aufwand getrieben haben, und auf deren Komplexität und intelligent gestellte Fallen wir besonders stolz sein dürfen.

Helfen wir also dem Spieler, der sowieso mehr als nur sieben Leben haben muß, um alle möglichen adventuregemäßen Arten des Ablebens kennenlernen zu können, und schaffen wir die technischen Voraussetzungen dazu, daß er das Spiel, um eine Erfahrung reicher, ab dieser Stelle fortsetzen kann. Wenn er dann nicht rechtzeitig Gebrauch von dieser sinnvollen Erweiterung macht, kann er sich nur über sich selber ärgern.

Außerdem verhindern wir mit diesen Routinen einen Schnitt ins eigene Fleisch, denn während der Testphase eines Adventures wird sich immer wieder die Notwendigkeit kleiner

Programmänderungen zeigen, welche leider auch die Inhalte aller Variablen zunichte machen.
Was bleibt, ist nur ein neuer Start mit Run und die Wiederholung aller notwendigen Eingaben, um wieder in den Raum zu gelangen, in dem der Fehler aufgetreten war, und um dann die Feststellung zu machen, daß die neue Version keineswegs eine Verbesserung bedeutet.

SAVE GAME / LOAD GAME

Überlegen wir zunächst, wodurch sich eine bestimmte Szene des Spieles von der Ausgangsstellung unterscheidet.

Sofort fällt uns der Spieler ein, der sich mehr oder weniger zielstrebig umherbewegt hat, ebenso denken wir an die Gegenstände, die er genommen oder abgelegt, zerstört, verbraucht oder gegessen hat, also an Gegenstände, die nicht mehr existieren ebenso wie an Dinge, die neu im Spiel aufgetaucht sind.

Neben diesen sichtbaren, und deshalb vielleicht weniger wichtigen, weil kontrollierbaren Variableninhalten, müssen wir ebenfalls zur Steuerung des Spielverlaufes gemachte Veränderungen retten, sonst kann es geschehen, daß eine Aktion trotz aller erforderlichen Gegenstände nicht den gewünschten Erfolg bringt.

Denken wir nur an die Flags, die nicht immer nur für den Verlauf einer zusammenhängenden Aktion, sondern auch für die Kontrolle scheinbar zusammenhangloser Ereignisse benutzt werden können.
Gleiches gilt für unsere Richtungstabelle, auch hier sind, wie wir später noch sehen werden, umfangreiche Veränderungen durchaus üblich.

Kurz gesagt: die Inhalte der Variablen **SPIELER**, **OB()**, **FL()** und **DURCHGANG(,)** müssen vor einem Spielabbruch als sequentielle Datei auf der Diskette bzw. Kassette gespeichert werden.

Exkurs: Externe Datenspeicherung

Neben der internen Datenspeicherung, bei der die Daten im eigentlichen Arbeitsspeicher der CPU abgelegt werden und ihr somit ständig zur Verfügung stehen, sind für den Umgang mit Daten, gleich welcher Art, auch externe Speicher notwendig, die den Speicher eines Computers fast beliebig erweitern und im wesentlichen der Archivierung und der Bereitstellung größerer Datenmengen dienen. Als Vertreter dieser Gattung sind beispielsweise Lochkarten und -streifen, Magnetbänder, Platten- und Walzenlaufwerke wie auch unsere Diskettenstation oder der in den CPC 464 eingebaute Datenrekorder zu nennen.

Abgesehen von verschiedenen Kapazitäten und Zugriffszeiten, lassen sich auch Unterschiede in der Art der Datenspeicherung erkennen.

Speichergeräte, wie ein Lochstreifenstanzer/leser oder wie der Datenrecorder, arbeiten dabei mit sogenannten sequentiellen Dateien, und sind am ehesten mit einer altertümlichen Papyrusrolle zu vergleichen, während ein Disketten- oder Plattenlaufwerk meist mit relativen Dateien arbeitet, was einem Karteikasten nahekommt.

Beide Verfahren haben ihre Vor- und Nachteile, die besonders im jeweiligen Platzbedarf und in der Zeit, die benötigt wird, um einen bestimmten Datensatz zu finden, zur Geltung kommen.
So wird jede Anschrift einer Kundendatei in einem Karteikasten eine eigene Karte beanspruchen, während auf

der Rolle zwecks Papierersparnis alle Adressen möglichst dicht notiert wurden. Wird aber die Adresse des Kunden 234 gesucht, nimmt man sich die 234. Karte, denn selbstverständlich sind alle numeriert, bei der Rolle wird es sich hingegen nicht vermeiden lassen, alle Anschriften zu lesen oder sie zumindest von vorne an durchzuzählen.

Die Leser, die sich nun mehr mit Dateien befassen möchten, mögen dies mit entsprechender Literatur tun, ansonsten mag die Bemerkung genügen, daß die sequentielle Speicherung für unser Problem genau das richtige Verfahren ist, denn wir wollen unsere Variableninhalte der Reihe nach speichern und später wieder laden, das heißt, besondere Zugriffsmethoden sind für uns nicht erforderlich.

Save Game

Die Übertragung der Daten wird zweckmäßigerweise innerhalb entsprechender Schleifen geschehen, wozu es jedoch nötig wird, die Kenndaten des Programmes um die Anzahl der verwendeten Flags zu erweitern, denn auch dieser Schleife muß ein Endwert zugrunde gelegt werden können.

Ebenso müssen wir eine Änderung der Programmzeile 1500 vornehmen, um die neuen Routinen in die Abfrage der Sonderbefehle einzubauen:

```
140 af=3

1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1600
```

Handelte es sich bei der letzten Eingabe des Spielers nicht um den Befehl Inventur, werden einige Sprünge ausgeführt, bis die Identität mit einem der Kurzbefehle festgestellt worden ist:

```
1599 REM ********************* SAVE GAME
1600 IF LEFT$(eingabe$,3)<>"SAV" THEN GO
TO 1700
1699 REM ********************* LOAD GAME
1700 IF LEFT$(eingabe$,3)<>"LOA" THEN GO
TO 1900
```

Bevor das Betriebssystem unseres Computers dann die Daten übertragen kann, müssen ihm zunächst einige Informationen, wie Übertragungsrichtung, Transportweg und Adressat, genannt werden, eine Aufgabe, die der Openbefehl in Zusammenarbeit mit der Print - Anweisung übernimmt.

```
1620 OPENOUT "SPIELSTAND"
```

Der Übertragung von Spieldaten steht nach dieser Vorbereitung nichts mehr im Wege. Die Ausgabe der Daten auf die Cassette erfolgt dabei in ähnlicher Form wie eine Ausgabe auf den Bildschirm, der Print Befehl muß nur durch die Angabe einer Filenummer (#9) umdirigiert werden.
Ein abschließender Close - Befehl übernimmt es, uns vor der Rückkehr ins Hauptprogramm vor später auftretenden Fehlern zu schützen.

```
930 FOR flag=1 TO af:fl(flag)=0:NEXT fla
g
1625 PRINT #9,spieler
1630 FOR objekt=1 TO ao
1631 PRINT #9,ob(objekt)
1632 NEXT objekt
1635 FOR raum=1 TO ar
1636 FOR richtung=1 TO 6
1637 PRINT #1,durchgang(raum,richtung)
1638 NEXT richtung
1639 NEXT raum
1645 FOR flag=1 TO af
1646 PRINT #9,fl(flag)
1647 NEXT flag
```

```
1650 CLOSEOUT
1670 GOTO 1080
```

Load Game

Zur Rekonstruktion einer bestimmten Situation ist es erforderlich, diese Daten wieder an die entsprechenden Variablen zu überweisen, eine Aufgabe, die ein fast identischer Programmteil übernimmt:

```
1720 OPENIN "SPIELSTAND"
1725 INPUT #9,spieler
1730 FOR objekt=1 TO ao
1731 INPUT#9,ob(objekt)
1732 NEXT objekt
1735 FOR raum=1 TO ar
1736 FOR richtung=1 TO 6
1737 INPUT #9,durchgang(raum,richtung)
1738 NEXT richtung
1739 NEXT raum
1745 FOR flag=1 TO af
1746 INPUT #9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1770 GOTO 1080
```

Um unser Programm noch luxuriöser auszustatten, werden wir noch einige weitere Zeilen ergänzen. So ist es durchaus nicht notwendig, daß wir uns auf ein einziges File zur Abspeicherung beschränken, eine zweite Datei wird sogar unbedingt erforderlich werden, wenn weitere Mitglieder unserer Familie ihr Herz für Adventures entdecken und sich an dem gleichen Spiel versuchen wollen.

Lassen wir den Spielern daher die freie Wahl und erlauben wir ihnen, einen Namen nach Wunsch, bis zu einer maximalen Länge von neun Buchstaben, zu wählen. Das Adventure soll dieses File dann mit der Ergänzung .ADVDAT als Spielstandsdatei kennzeichnen.

Als zweckmäßig erweist es sich weiterhin, den Abbruch des Programmlaufes im Falle eines auftretenden Fehlers zu verhindern, was sehr leicht geschehen kann, wenn eine falsche Cassette (oder Diskette) in das betreffende Laufwerk eingelegt wurde.
Solchermaßen komplettierte Unterprogramme werden dann ähnlich den folgenden Listings aussehen.

```
1599 REM ********************** SAVE GAME
1600 IF LEFT$(eingabe$,3)<>"SAV" THEN GO
TO 1700
1610 PRINT"REC & PLAY druecken ...":PRIN
T"Unter welchem Namen speichern ........
..";STRING$(10,8);:INPUT eingabe$
1620 IF LEN(eingabe$)>10 THEN PRINT"Bitt
e etwas kuerzer !":GOTO 1610 ELSE eingab
e$="!"+eingabe$+".SPIEL":OPENOUT eingabe
$
1625 PRINT£9,spieler
1630 FOR objekt=1 TO ao
1631 PRINT£9,ob(objekt)
1632 NEXT objekt
1635 FOR raum=1 TO ar
1636 FOR richtung=1 TO 6
1637 PRINT£9,durchgang(raum,richtung)
1638 NEXT richtung
1639 NEXT raum
1645 FOR flag=1 TO af
1646 PRINT£9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1660 PRINT ok$
1670 GOTO 1080
```

```
1699 REM *********************** LOAD GAME
1700 IF LEFT$(eingabe$,3)<>"LOA" THEN GO
TO 1900
1710 PRINT"Cassette rueckspulen & PLAY d
ruecken ...":PRINT"Welches Spiel laden .
.........";STRING$(10,8);:INPUT eingabe$
1720 IF LEN(eingabe$)>10 THEN PRINT"Das
kann nicht sein !":GOTO 1710 ELSE eingab
e$="!"+eingabe$+".SPIEL":OPENIN eingabe$
1725 INPUT£9,spieler
1730 FOR objekt=1 TO ao
1731 INPUT£9,ob(objekt)
1732 NEXT objekt
1735 FOR raum=1 TO ar
1736 FOR richtung=1 TO 6
1737 INPUT£9,durchgang(raum,richtung)
1738 NEXT richtung
1739 NEXT raum
1745 FOR flag=1 TO af
1746 INPUT£9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1760 PRINT ok$
1770 GOTO 1080
1780 PRINT"Achtung Fehler !":BORDER 3:RE
SUME NEXT
```

DER WORTSCHATZ

Häufig macht die richtige Wortwahl ein Adventurespiel zur Qual, denn schließlich wird man als Spieler gezwungen, sich der gleichen Ausdrucksweise wie der Programmierer zu bedienen, was nicht immer ganz einfach ist.
Importe, die für den fortgeschritteneren amerikanischen Markt bzw. englischen produziert wurden, lassen sich mit dem Schulenglisch häufig gar nicht, fast immer aber nur schwer lösen. Ein englisch - deutsches Wörterbuch hat bei einigen Spielern daher seinen festen Platz in Reichweite des Computers, doch auch dieses Hilfsmittel versagt, wenn sich der Autor der Umgangssprache oder eines Slangs bedient hat.

Scheitern viele Spiele an der Sprachbarriere, freut man sich um so mehr, wenn aus irgendeiner Quelle plötzlich ein deutsches Adventure auftaucht.
Doch sofort trüben zwei Eigenarten den rosaroten Blick, denn unsere Muttersprache ist bei weitem nicht so gut dazu geeignet, ähnlich wie bei den englischsprachigen Originalen, mit nur zwei Worten die unterschiedlichsten Situationen zu artikulieren. Schlimmes Gastarbeiterdeutsch mag noch akzeptabel sein, und auch über die unmöglichsten Satzkonstruktionen kann man zumindest zu Beginn noch lachen, doch wie soll ein normal begabter und veranlagter Mensch ohne Hilfestellung auf diese Eingaben kommen ?

´Nimm Messer´ ist dabei gar nicht einmal so übel, doch das Messer wird auch eine Aufgabe zu erfüllen haben. Es bleibt jedoch weiterhin ein erfreulicher Gegenstand, denn Eingaben wie ´Wirf Messer´, ´Schleife Messer´ oder ´Schärfe Messer´ erfordern nicht allzuviel Phantasie und sind ebenfalls recht eindeutig auszulegen.

Hat das Messer seine Funktion erfüllt, muß man sich seiner früher oder später entledigen, was bei einigen Spielen durch bewußtes Verlieren geschehen kann. Etwas besser als

´Verliere Messer´ wirkt schon ´Leg Messer´, doch taucht automatisch die Frage nach dem wohin auf oder es wird zumindest noch ein ´weg´ erwartet. Diese Eingabe steht zwar im Widerspruch zu den üblichen Spielregeln, sollte aber dennoch, zumindest auf unserem System, versucht werden.

Betrachten wir abschließend noch einmal ein Beispiel der ersten Seiten dieses Buches. Wie gelangen wir in die Krone des Baumes, vielleicht mit ´Kletter Baum´ ?

Leider nicht, aber wohin bringt uns ein besser klingendes ´Erklettere Baum´ ?

Wieder nur die Mitteilung, daß das Wort ´Erklettere´ nicht verstanden wurde, also aufgeben oder weitergrübeln ?
Es wird sich nicht vermeiden lassen, Verben wie ersteige, besteige und erklimme zu erproben, und wenn die Aktion wirklich erforderlich ist, wird der Erfolg sich früher oder später einstellen.

Probleme dieser Art sind natürlich nicht sehr schön, und da dieses Kapitel von der Perfektionierung unseres Adventuresystemes handelt, werden wir uns nach geeigneten Lösungsmöglichkeiten umsehen.

SYNONYME

Gleichbedeutende Worte in reichlicher Menge einzuarbeiten, wäre ein erster Vorschlag zur Lösung des Sprachproblemes, wie jedoch das Beispiel des Baumes zeigt, kann die konsequente Ausstattung mit gleichbedeutenden Worten den Umfang eines Programmes sehr vergrößern, was ein immenses Maß zusätzlicher Routinearbeit bedeutet und den sowieso immer knapper werdenden freien Speicherplatz stark einschränkt.

Dennoch soll hier, wieder am Beispiel unserer Miniversion, von Goldrausch, ein Lösungsvorschlag eingebracht werden, der durch seine Kürze und Einfachheit doch schon fast wieder überzeugt. Selbstverständlich handelt es sich um keine zwingend erforderlichen Programmzeilen, es liegt an Ihnen, ob Sie Wert auf einen großen Wortschatz legen oder ob Sie sich mit dem anschließend gemachten Vorschlag zur Verbesserung der Benutzerfreundlichkeit zufrieden geben wollen.

Die erste Gelegenheit, nicht sofort das richtige Wort zu treffen, hat ein Spieler von Goldrausch in Raum 3 bei dem Versuch, die Holzhütte näher in Augenschein zu nehmen. Bislang wird er mit ´Untersuche Holzhütte´ scheitern, der Erfolg wird sich erst bei ´Untersuche Hütte´ einstellen.

Damit der zweite Begriff überhaupt verstanden werden kann, müssen wir unser Vokabular zunächst erweitern:

```
120 ao=20
320 DATA "-","holz",0
```

Die ausführliche Beschreibung können wir uns sparen, desgleichen wird die Null als Aufbewahrungsort verbindlich, schließlich soll weder im Inventory noch in einem Raum eine weitere Hütte auftauchen.
Benötigt wird nur der Rufname, damit der Adventuretreiber die entsprechende Objektnummer ermitteln kann, was in Zeile 2140 auch geschieht.

Normalerweise wird die Ausführung des Programmes in Zeile 2200 mit einem Sprung zur jeweiligen Befehlsausführung fortgesetzt. Dort findet sich natürlich keine einzige Bedingung, die erfüllt wäre, da nirgendwo für o ein Wert von 20 gefordert wird. Um diese Zeilen nicht ergänzen zu müssen, fügen wir folgenden Kunstgriff ein,

```
2140 IF eobjekt$=rufname$(o) THEN 2170
2165 REM ********************** SYNONYME
2170 IF o=20 THEN o=4
```

und schon werden mit der Holzhuette genau die gleichen Aktionen durchgeführt wie mit der Huette.

Um unser Programm in die Lage zu versetzen, auch eine Vielzahl von Verben verstehen zu können, ist der zu treibende Aufwand sogar noch geringer.
Ein die gleiche Aktion einleitendes Wort wird ebenfalls dem Wortschatz des Adventures zur Verfügung gestellt, und die Sprungtabelle wird einfach um das bereits einmal benutzte Sprungziel ergänzt.

Goldrausch erweitern wir auf diese Weise um die zu ´Untersuche´ identische Funktion ´Betrachte´:

```
130 av=7
202 DATA betr
2200 ON verbnummer GOTO 5000,6000,7000,8
000,9000,10000,5000
```

Sie sehen, daß unser System sich um gleichbedeutende Verben besonders einfach erweitern läßt, weshalb wir zumindest auf diesen Bonus nicht verzichten sollten.

Synonyme der Objekte dürfen wir stiefmütterlich behandeln, denn ihnen kommt sowieso nicht die gleiche Bedeutung zu, denn die Kopfzeilen des Spieles werden fast immer, mit Ausnahme bei den Grafikadventures, Anhaltspunkte geben.

Wir können es dem Spieler aber auch einfacher machen, so daß er sich wirklich auf die Problemlösungen konzentrieren kann und auf der Suche nach den richtigen Worten nicht den roten Faden verliert.

Was würden Sie davon halten, wenn ein Adventure auf Wunsch sein gesamtes Vokabular preisgibt, wenn es eine Liste aller vorhandenen Objekte und möglichen Verben erstellt ?

Ich verstehe folgende Verben:

untersuche
nimm
leg
oeffne
benutze
zerstoere
zuende
fuelle
betrete
loesche
befestige

Taste druecken

Ein solches Bild dürfte alle Probleme der Wortwahl aus der Welt schaffen, und der Spieler verliert nicht durch sinnlose Eingaben Zeit und Lust.

Leider läßt sich an dieser Stelle zum zweiten Mal der Volksmund zitieren, denn ´Wo Licht ist, ist auch Schatten´.

Stellen wir uns einen Spieler vor, dem Goldrausch noch nicht bekannt ist. Ohne viel gespielt zu haben, wird er, wenn er einigermaßen konsequent und logisch denken kann, sogleich eine Lösungsstrategie entwickeln, die ihn schnell ans Ziel bringt. Er kennt sofort den gesamten Wortschatz und weiß, daß sich irgendwo eine Flasche finden läßt, daß irgendwo eine Truhe liegt und wird ebenfalls sogleich ein starkes Verlangen nach Silber und Nuggets entwickeln. Anschließend überlegt er, welche Gegenstände sich zerstören

lassen, und seine Wahl wird auf die Flasche, die Hütte und die Kette fallen.
Er bemüht sich, die Schatztruhe zu finden, und nachdem er die Feststellung machte, daß eine Eisenkette ihn am Sichten des Inhaltes behindert, versucht er diese zu zerstören, was aber mit bloßen Händen kaum gelingen wird. Ein weiteres VOK wird ihm dann die Ahnung vermitteln, daß die Eisenstange vermutlich der einzige Gegenstand ist, der für diese Aufgabe in Frage kommt.

Aber auch die Gefahren werden entschärft, denn es bleibt ihm auch nicht verborgen, daß außer ihm noch ein Bär die Welt bevölkert !

Somit wird dem Programmierer auch noch die Aufgabe zufallen, genau abzuwägen, ob er den gesamten Wortschatz des Adventures preisgeben will oder nur eine Untermenge.

Dies könnte problemlos durch ein weiteres Titelbild geschehen, welches eine Anzahl der wichtigsten Worte ausgibt, allerdings würden wir damit unserem Ziel, ein universelles Adventuresystem zu entwickeln, entgegenhandeln, weil diese Tafel für jedes Programm neu erstellt werden müßte.

Für große Adventures mit einem entsprechenden Wortschatz bleibt die Auflistung des Vokabulars jedoch durchaus sinnvoll, weshalb wir bei der Entwicklung unseres Adventuresystemes ebenfalls nicht darauf verzichten wollen.

Die Behandlung des VOK - Befehles übernimmt aus eben genanntem Grunde der Treiber. Um eine wirklich universelle Verwendung dieses Programmteiles zu gewährleisten, müssen die zur Ausgabe bestimmten Worte aus den Spieldaten des betreffenden Adventures ermittelt werden.

Es wurde bereits kurz erwähnt, daß der Basic - Befehl Restore den sogenannten DATA - Zeiger auf den Beginn des Datenblockes stellt, und das nächste Read wieder das erste Element der Liste einer Variablen zuweisen wird.

Glücklicherweise kommt die Anordnung unserer Adventuredaten einem solchen Vorgehen entgegen, da zuerst die Verben und daran anschließend die Objekte genannt werden, es muß keine Ausführungszeit auf das Lesen der übrigen Daten verwandt werden.

Bauen wir das entsprechende Unterprogramm, unter Berücksichtigung der wahrscheinlichen Häufigkeit eines Aufrufes der entsprechenden Zeilen, im Anschluß an die INV, SAVE, und LOAD Routine ein:

```
1700 IF LEFT$(eingabe$,3)<>"LOA" THEN GO
TO 1800
1800 IF LEFT$(eingabe$,3)<>"VOK" THEN GO
TO 1900
1805 CLS:PRINT"Ich verstehe folgende Ver
ben:":PRINT:PRINT:RESTORE
```

Ein nachfolgendes Read verhält sich nun genau so, wie der erste Read - Befehl nach dem Einschalten des Computers. Übernehmen wir daher die entsprechenden Programmzeilen aus Teil 1 mit der Änderung, daß die eingelesenen Daten nicht weiter gespeichert, sondern nur auf dem Bildschirm ausgedruckt werden:

```
1810 FOR i=1 TO av
1820 READ wort$:PRINT wort$
1830 NEXT i
```

Anschließend wollen wir dem Spieler ausreichend Zeit geben, die Verbliste zu studieren. Da Gleiches auch nach Ausgabe der Objekte geschehen soll, realisieren wir die entsprechenden Zeilen als Unterprogramm.

Dieses wird dann an den betreffenden Stellen mit Gosub aufgerufen:

```
1840 GOSUB 1890
1845 CLS:PRINT"und folgende Gegenstaende
sind mir":PRINT"bekannt":PRINT
1850 FOR i=1 TO ao
1860 READ wort$,wort$,x:PRINT wort$
1870 NEXT i
1880 GOSUB 1890:CLS:GOTO 1080

1890 LOCATE 1,25:PRINT "Taste drucken"
1895 eingabe$=INKEY$:IF eingabe$="" THEN
 1895 ELSE CLS:RETURN
```

Zu beachten ist, daß ohne die Verwendung entsprechender Basic - Erweiterungen, DATA - Zeilen ebenfalls sequentiell gelesen werden. Zeile 1860 weist der Stringvariablen wort$ daher zunächst die ausführliche Objektbeschreibung zu, die, weil wir diesen langen Text an dieser Stelle nicht benötigen, daran anschließend sofort mit dem Rufnamen überschrieben wird.

Zum Abschluß der Auflistung des gesamten Vokabulars wird wieder das Unterprogramm ab Zeile 1890 aufgerufen, damit nach Drücken einer weiteren Taste das Adventure fortgesetzt werden kann.

Obige Zeilen werden nun wie vorgesehen arbeiten, zumindest solange nicht mehr als zwanzig Worte ausgegeben werden müssen. Nach überschreiten dieser Zahl wird die Verweildauer der ersten Vokabeln auf dem Bildschirm kaum von ausreichender Länge sein, um eine Hilfe für den Spieler zu sein.

Zur Lösung des Problemes führen wir die Kontrollvariable Zeile ein. Sie wird zu Beginn der Ausgabe auf Null gesetzt,

und bei der Ausgabe eines jeden Wortes um eins erhöht. Sind mit zwanzig Worten entsprechend viele Zeilen benutzt worden, löschen wir den Bildschirm und setzen den Zähler wieder auf eins zurück.

```
1849 zeile=0
1850 FOR i=1 TO ao
1855 zeile=zeile+1
1860 READ wort$,wort$,x:PRINT wort$
1865 IF zeile=20 THEN GOSUB 1890
1866 IF zeile=20 THEN zeile=1:CLS
1870 NEXT i
```

Ein Programmlauf bringt nun das gewünschte Ergebnis, doch entsprechen die erscheinenden Kürzel UNT, OEF und wie sie alle heißen keineswegs dem Standard unserer Programme.

Um ein zufriedenstellenderes Ergebnis zu erzielen, ändern wir die Verben und Objekte in den für sie vorgesehenen DATA - Zeilen (200 - 500) und machen uns die Mühe, sie, ohne Berücksichtigung der relevanten Wortlänge, auszuschreiben.

Durch diese Manipulation ergibt sich nun die Notwendigkeit einer weiteren Ergänzung in den Zeilen 815 und 835, schließlich wollen wir den für die Variablen vorgesehenen Speicherplatz nicht unnötig verschwenden.

Warum soll das ganze Wort gespeichert werden, wenn zur eindeutigen Identifizierung eines Spielzuges nur drei oder vier Buchstaben von Bedeutung sind.

```
815 READ verb$(i):verb$(i)=LEFT$(verb$(i
),wortlaenge):verb$(i)=UPPER$(verb$(i))
835 READ ob$(objekt),rufname$(objekt),ob
(objekt):rufname$(objekt)=left$(rufname$
(objekt),wortlaenge):rufname$(objekt)=UP
PER$(objekt)
```

HELP

Die Entwicklung einer Help - Routine soll nun ein letzter Schritt sein, um die Benutzerfreundlichkeit unserer Adventures zu erhöhen.

Glaubt der Spieler, daß er sich restlos verfahren hat, so wird er versuchen, durch die bloße Eingabe von Help in den Besitz hilfreicher Informationen zu gelangen.

Handelt es sich um ein einfaches Spiel, weil im Verlauf desselben genügend Hinweise gegeben werden, machen wir uns die Sache einfach:

```
1959 REM **************************** HELP
1960 IF LEFT$(eingabe$,4)<>"HELP" THEN G
OTO 2000
1970 PRINT"Ich kann die Spielregeln wied
erholen.":PRINT"Wuenschen Sie das ?"
1971 eingabe$=INKEY$:IF eingabe$="" THEN
1971 ELSE GOSUB 900
1975 GOTO 1080
1979 REM ********************* ENDE HELP
```

Eine noch häufiger anzutreffende Standardantwort lautet ´Ich würde alles untersuchen ! - Try examining things !".
Meist werden jedoch, zumindest in den Abenteuerspielen neueren Datums, Ratschläge gegeben, die manchmal allerdings wenig hilfreich, weil für den Spieler nicht zu verstehen, sind.

Denn da es sich um einen Ein - Wort Befehl handelt, steht die gewünschte Hilfe nicht in Beziehung zu einem bestimmten Objekt, was eine Auswertung der Situation in Hinblick auf die Schwierigkeiten des Spielers nicht gerade einfach macht.

Schließlich steht dem Programm als Arbeitsgrundlage nur der Aufenthaltsraum des Spielers zur Verfügung, und so ist es nicht weiter verwunderlich, daß viele Adventures in problematischen Räumen immer wieder die gleiche Mitteilung an den Abenteurer ausgeben, die er deshalb nicht verstehen kann, weil seine Gedanken bislang um ein anderes Problem kreisen.

Als typisches Beispiel ziehen wir eine Spielszene aus Raum 4 heran. Der Spieler hat die Schatztruhe entdeckt und müht sich seit längerer Zeit vergeblich mit der hinderlichen Eisenkette ab. In dieser Situation wäre folgender Hinweis denkbar:

```
1970 IF spieler=4 THEN PRINT"Ein verlaen
gerter Arm verstaerkt die":PRINT"Kraft !
":GOTO 1080
```

Was könnte ein Spieler, der in Raum 4 um Hilfe bittet, mit der Information, an die Hebelgesetze zu denken, anfangen, wenn er bereits bei der Suche nach der Truhe gescheitert ist und er somit die Kette noch nicht entdeckt haben kann ?

Will man den Spieler verwöhnen und in besonders schwierigen Situationen Schritt für Schritt an die Lösung heranführen, wird es, wie obiges Beispiel zeigt, nicht ohne die Formulierung umfangreicher Tests und Bedingungen gehen. Für diese Aufgabe scheinen wieder einmal die Flags wie auch die Aufbewahrungsorte der verschiedensten Gegenstände prädestiniert zu sein.

Wiederum am Beispiel der Truhe probieren wir einmal diese Art der genau dosierten Hilfestellung aus, was allerdings nicht heißen soll, daß diese Szene als besonders schwierig zu gelten hat:

```
606 m$(6)="Wie kann ich die Kette zersto
```

```
eren ?"
607 m$(7)="Sie verhindert ein Oeffnen de
r Truhe !"

1970 IF spieler=4 AND ob(10)=0 THEN PRIN
T"Fast waere ich in ein Loch gefallen.":
GOTO 1080
1971 IF spieler=4 AND ob(11)=spieler AND
NOT fl(2) THEN PRINT m$(6):PRINT m$(7):G
OTO 1080
```

Erinnern wir uns, daß beim Passieren des betreffenden Raumes zunächst nur die Büsche sichtbar sind, und das Erdloch zwischen den Büschen erst entdeckt werden muß. Erlangt der Spieler jedoch die Information eines Beinahe - Sturzes in eine Grube, wird er sich wohl näher mit diesem Hindernis beschäftigen.

Ein weiterer Versuch mit Help bringt zu diesem Zeitpunkt keinen besonderen Vorteil. Erst nachdem die Truhe als sichtbares Objekt zur Ausstattung des Raumes gehört, erhält der Spieler den Hinweis, daß die Begutachtung des Inhaltes die Zerstörung der Kette und das Öffnen des Deckels erfordert.

Diese Aktionen wird er dann wohl hoffentlich alleine durchführen können, falls nicht, wäre es für diesen Spieler eine Überlegung wert, ob er nicht lieber einen Roman lesen oder es mit Arcade - Action Spielen versuchen sollte.

Den Abschluß einer solchermaßen durchkonstruierten Hilfestellung wird wieder eine Sicherheitszeile, wie wir sie bereits bei der Programmierung der anderen Aktionen kennengelernt haben, bilden, eine Zeile, die immer dann ausgeführt wird, wenn keine speziell definierte Situation eingetreten ist:

```
1971 ...
1972 ...
1975 PRINT"Erst sehen, dann denken und z
uletzt handeln !":GOTO 1080
```

MOTIVIERUNG DES SPIELERS

Unser nächster Vorschlag für bessere Adventures wird unseren Adventuretreiber um einen letzten Ein - Wort Befehl erweitern.

Abenteuerprogramme benötigen für ihre Lösung nun einmal einen gewissen Zeitaufwand, wenn man jedoch tagelang spielt und überhaupt keinen Fortschritt sieht, fragt man sich eines Tages doch, ob man nicht ein anderes Spiel laden soll.

Aus diesem Grunde halte ich Adventures, die mit einer Punktwertung arbeiten und dem Spieler anzeigen, wie weit er noch von seinem endgültigen Ziel entfernt ist, für bedeutend reizvoller.

Jedoch gilt es zu überlegen, auf welcher Basis die Wertung durchgeführt werden soll. Die übliche Version sieht für jeden gefundenen Schatz beziehungsweise für jede gelöste Teilaufgabe in einem Mission - Adventure eine gewisse Anzahl von Punkten vor. Neben der Angabe ´Von 100 möglichen Punkten hast du 40´ wird dem Spieler auch noch der prozentuale Anteil mitgeteilt, wodurch der Spielstand einfacher und genauer ausgewertet wird. Als Beispiel kann wieder einmal Scott Adams ´Adventureland´ angeführt werden, dreizehn Schätze ergeben eine krumme Anzahl von Punkten, bei mehr als 50 Prozentpunkten weiß der Spieler jedoch, daß er die Hälfte der Strecke geschafft hat.

Eine alternative Lösung wird dagegen jeden Schritt voran belohnen und auch nicht davor zurückschrecken, für jede geleistete Hilfe eine gewisse Anzahl von Punkten abzuziehen. Ebenso wird jeder Fehler mit Todesfolge negativ zu Buche schlagen, so daß im extremen Fall sogar ein negativer Wert erzielt werden kann. Doch ist es gerade dieser große Spielraum, der zu wiederholten

Lösungsversuchen reizt, denn es müßte doch möglich sein, mehr Punkte zu erzielen als der Freund, der das Ziel ebenfalls erreicht hat.

Soll ein Adventure dermaßen ausgestattet werden, beginnt die Arbeit des Programmierers mit der Auswahl geeigneter Sequenzen, schließlich soll nicht bereits das einfache Auffinden eines Gegenstandes Punkte wert sein. Eine Belohnung darf der Spieler jedoch erwarten, wenn er gefährliche Monster ausgetrickst oder geheime Durchgänge gefunden hat.

Bevor wir uns nun detailliert mit der praktischen Ausführung entsprechender Versionen befassen, vermitteln wir als erstes dem Treiber das Verständnis für den Befehl ´SCORE´:

```
1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1560
1559 REM *************************** SCORE
1560 IF LEFT$(eingabe$,3)<>"SCO" THEN GO
TO 1600
```

Anschließend führen wir zur Speicherung des erzielten Punktewertes die Variable WERTUNG ein. Dabei ist es wichtig, dafür Sorge zu tragen, daß sie bei jedem Neustart, somit auch beim Tode der Hauptfigur, auf Null gesetzt wird.

Nur der erreichte Stand des Punktekontos sagt natürlich gar nichts aus, wenn er nicht in Beziehung zur maximalen Punktzahl steht, deshalb:

```
170 wmax=20 :REM maximal moegliche Punkt
zahl der Miniversion
```

```
171 wertung=0 :REM bis jetzt 0 Punkte
1561 PRINT"von";wmax;"Punkten hast Du";w
ertung;"Punkte."
1565 GOTO 1080
```

Die Punkte erzielt der Spieler mit dem Aufnehmen der Gold- und Silberstücke. Ergänzen wir darum die betreffenden Zeilen in unserem Aktionsteil:

```
6014 IF o=12 THEN PRINT ok$:ob(12)=-1:we
rtung=wertung+10:GOTO 1080
6017 IF o=17 AND fl(3) THEN PRINT ok$.:o
b(17)=-1:wertung=wertung+10:GOTO 1080
```

Leider kann ein Spieler, der es nur auf Punkte abgesehen hat, mit der derzeitigen Ausführung des Programmes beliebig hohe Summen erlangen.

Erinnern wir uns, daß wir bei der Ausgestaltung der Aktion ´Nimm´ darin übereingekommen waren, daß mit nehmen ein ´in die Hand nehmen´ gemeint war. Dies ermöglichte der Spielpraxis neben der Aufnahme von Gegenständen zwecks Aufbewahrung im Inventory auch ein Nehmen aus dem Inventory, um eine Aktion, beispielsweise das Öffnen einer Tür mittels eines Schlüssels, durchzuführen.

Gerade diese Auslegung erweist sich nun als hinderlich, nichts hält den Spieler davon ab, mehrmals ´Nimm Silbermuenzen´ einzugeben und mit dieser Eingabe jeweils zehn Punkte auf seinem Konto hinzuzuaddieren.

Als Ausweg bieten sich einmal eine Änderung der allgemeinen Vorbedingung (Zeile 6000) wie auch die Einführung zusätzlicher Bedingungen in die Aktion selbst, an.
Dieses Prinzip kennen wir bereits von der Ausführung der Aktion ´Untersuche´, hier mußte eine ähnliche Regelung

verhindern, daß ein Gegenstand immer wieder an seinem Ursprungsort sichtbar wurde.

```
6014 IF o=12 AND ob(o)=4 THEN PRINT ok$:
ob(12)=-1:wertung=wertung+10:GOTO 1080
6017 IF o=17 AND ob(o)=5 AND fl(3) THEN
PRINT ok$:ob(17)=-1:wertung=wertung+10:G
OTO 1080
6020 IF o=12 AND ob(o)=-1 THEN PRINT"Das
Silber habe ich bereits !":GOTO 1080
6021 IF o=17 AND ob(o)=-1 THEN PRINT"Ich
bin bereits im Besitz des Goldes !":GOTO
 1080
```

Entscheiden Sie sich hingegen, die Aktion ´Nehmen´ immer nur als Bereicherung des Spielers aufzufassen, können Sie sich auf eine Verschärfung der Eingangsbedingung und Ausgabe eines entsprechenden Hinweises beschränken:

```
6000 IF ob(n)<>spieler THEN GOTO 6900
6998 IF ob(n)=spieler THEN PRINT"Das hab
e ich doch schon !":GOTO 1080
```

Für unsere eigenen, das heißt zumindest für die Adventures dieses Buches, wollen wir nur eine Punktierung der in den Besitz des Spielers gebrachten Schätze vornehmen. Dennoch werden wir nicht auf eine Bewertung der einzelnen Züge des Spielers verzichten müssen.

Im Gegenteil, wir werden sogar noch einen Schritt weiter gehen und jeden einzelnen Spieler bewerten, ihm die erreichte Durchschnittswertung mitteilen.

Als Grundlage für diese Auswertung bietet sich die Anzahl der benötigten Züge an. Ein erfahrener Abenteurer wird sich

nicht mit der Untersuchung jeder Kleinigkeit aufhalten, ebenso weiß er, aus typischen Hinweisen seinen Nutzen zu ziehen.
Aus dieser Summe und den dabei erreichten Punkten berechnen wir einen Durchschnittswert und nehmen anhand dieses Quotienten die Beurteilung des Spielers vor.

```
180 zug=0:wertung=0

1080 PRINT:PRINT:zug=zug+1

1561 PRINT"von";wmax;"Punkten hast Du in
";zug;"Zuegen"
1562 PRINT wertung;"Punkte erreicht !":P
RINT"Das entspricht einem Schnitt von"
1563 PRINT wertung/zug;"Punkten !"
1565 GOTO 1080
```

Auch das Kriterium für ein erfolgreiches Spielende definieren wir neu. Anstelle einer genau definierten Bedingung, die spielabhängig ist, vergleichen wir nun die erreichte Punktzahl mit dem maximal möglichen Wert. Ist die Differenz gleich Null, hat der Spieler die Aufgabe gelöst.

Zweckmäßigerweise erfolgt dieser Test gleich zu Beginn eines jeden Zuges, bevor überhaupt irgendwelche Daten an den Spieler übermittelt werden.

```
1085 IF wertung=wmax THEN GOTO 4800
```

Falls Sie ebenfalls die Punktzahl als Kriterium verwenden, vergessen Sie bitte nicht, alle anderen Zeilen, die mit der gleichen Aufgabe beschäftigt sind, aus dem Programm zu entfernen (Zeilennummern im Bereich von 1331 bis 1389).

Mit der konsequenten Einführung eines Punktesystems stellt sich uns zum Abschluß noch die Aufgabe, den Siegestitel an

diesen Aufbau anzupassen. Die ermittelten Daten sollten noch einmal zusammengefaßt werden, damit auch eine Benotung des Spielers stattfinden kann.

Dazu wird es sich allerdings nicht vermeiden lassen, daß Sie, wenn das Adventure fertiggestellt ist und keinerlei Fehler mehr aufweist, einen kompletten Lauf durchspielen. Bei diesem Spiel verzichten Sie auf alle nicht unbedingt erforderlichen Eingaben und ermitteln die Anzahl der für einen Sieg mindestens erforderlichen Züge. Berechnen Sie das Verhältnis zwischen Punkte und notwendigen Eingaben und legen Sie diesen oder einen etwas geringeren Wert als Kennzeichen für ein sehr gutes Spiel zugrunde.
Wählen Sie entsprechend abgestufte Zahlen für schlechtere Beurteilungen.

So kann die Benotung eines Spielverlaufes der Miniversion von Goldrausch etwa folgendermaßen vorgenommen werden:

```
4800 CLS
4805 endwertung=wertung/zug
4810 PRINT"Herzlichen Glueckwunsch !"
4815 PRINT"Sie haben die Ihnen gestellte
Aufgabe"
4820 PRINT"geloest."
4825 PRINT"In ";zug;" Spielzuegen haben
Sie pro Zug"
4830 PRINT endwertung;" Punkte gemacht."
4835 PRINT"Damit haben Sie ein ";
4840 IF endwertung<.5 THEN m$(0)="misera
bles"
4845 IF endwertung>0.5 THEN m$(0)="maess
iges"
4850 IF endwertung>1.5 THEN m$(0)="sehr
gutes"
4855 IF endwertung>1.0 THEN m$(0)="gutes
"
4860 PRINT m$(0);" Ergebnis"
```

```
4865 PRINT"erzielt."
4899 END
```

Ein solchermaßen gestalteter Titel, entsprechen aufgemacht durch die Verwendung von Steuer-, Farb- und Grafikzeichen, wird manchen Abenteurer, der das Adventure endlich gemeistert hat, dazu veranlassen, es noch einmal zu spielen.

Und daß alles nur, um als guter Spieler bezeichnet zu werden.

Mit diesen Bemerkungen wäre das Thema ´Score´ eigentlich beendet, wenn wir nicht noch eine technische Ergänzung machen müßten, deren Fehlen ansonsten die ganze schöne Planung zunichte machen kann und mit Sicherheit die Spieler unserer Programme verärgern würde.

Vielleicht haben Sie bereits an unsere Save Game Routine gedacht, der natürlich auch ein Abspeichern und Laden der oben eingeführten Daten Wertung und Zug ermöglicht werden muß, wenn der Spieler nicht immer wieder mit null Punkten beginnen soll.

```
1627 PRINT#9,wertung
1628 PRINT#9,zug

1727 INPUT#9,wertung
1728 INPUT#9,zug
```

Alle bisher in diesem Kapitel gemachten Vorschläge dienten einzig und allein dem Ziel, den Komfort der Programme zu erhöhen.

Ich glaube, daß Sie mit mir einer Meinung sind, wenn ich die Behauptung aufstelle, daß Adventures, die nach dem Konzept dieses Buches entwickelt wurden und sämtliche Extras enthalten, durchaus zur Oberklasse der Textadventures gezählt werden dürfen. Schließen wir daher die technische Entwicklung mit diesen Zeilen ab, und wenden wir uns der Frage zu, wie wir die Spiele selbst attraktiver und auch schwieriger machen können.

Dabei werden wir es halten wie gewohnt, nach trockener Theorie folgen entsprechende Programmzeilen, die wiederum für unser Erstlingswerk Goldrausch gedacht sein werden.

Die folgenden Seiten werden Ihnen insbesondere zeigen, wie Sie dem Spieler weitere Steine in den Weg legen können, so daß er tatsächlich die angebotenen Kurzbefehle Save und Load nutzen muß, um irgendwann einmal ans Ziel zu gelangen.

Um ausreichend Platz für Monster, Falltüren und was es sonst noch gibt, zu haben, werden wir uns als nächstes wieder einmal mit der Geografie befassen und unsere Miniwelt zur Grundlage eines ausgewachsenen Adventures entwickeln.

Falls Sie allerdings beabsichtigen, selber in den Genuß eines Adventurespieles zu kommen, sollten Sie, bevor Sie sich mit den nächsten Abschnitten befassen, alle noch fehlenden Programmzeilen eingeben oder besser noch, von jemand anderem eingeben lassen.

Das komplette Listing von Goldrausch, in welchem Sie Beispiele zu allen Vorschlägen finden, entnehmen Sie bitte dem letzten Kapitel.

ORIENTIERUNG ODER DESORIENTIERUNG

Es wurde bereits zu Beginn des Buches erwähnt, daß die meisten Abenteuerspiele sich innerhalb einer Welt von ungefähr vierzig Räumen abspielen. Dennoch entsteht der Eindruck einer viel größeren Welt, gab es da doch den riesigen Wald, einen Sumpf, eine unterirdische Höhlenlandschaft, ein Labyrinth und und und .

Tatsächlich sind jedoch gerade die so weiträumig erscheinenden Gebiete Gruppen von einigen wenigen Räumen, die auf so geschickte Art und Weise miteinander verbunden wurden, daß der Spieler meistens nicht einmal bemerkt, auf welche Tour er da geschickt wurde.

Die Palette der Möglichkeiten reicht dabei vom einfachen Raum, der immer wieder in sich selbst mündet, bis zu einer Kette von Räumen, an deren Ende der Spieler wieder an den Beginn versetzt wird.

Nutzen wir an dieser Stelle die in Goldrausch ähnlichen Räume 1 und 2, um gleich zu Spielbeginn ein weitläufiges Areal zur Verfügung zu haben.
Lassen Sie Raum 1 in nördlicher und westlicher Richtung wieder in sich selbst zurückmünden, verfahren Sie gleichermaßen mit Raum 2, und versuchen Sie dann, sich den Spieler vorzustellen, der in Raum 1 beginnt und nach Westen wandert. Woher soll er wissen, daß er sich nach jeder Eingabe immer noch im gleichen Teil des Waldes befindet ?
Falls er nicht das Glück hat, das Spiel mit der Eingabe ´O´ zu beginnen, wird er eine Reihe von Eingaben benötigen, um den Weg aus dem Wald zu finden, was ihm einen guten Schnitt bei der Punktwertung bereits verdirbt.

```
501 ... 1,1,1,2,0,0
502 ... 2,2,3,1,0,0
```

Noch schwieriger wird sich die Suche nach dem rechten Weg gestalten, wenn der südliche Ausgang von Raum 2 den Spieler nicht wieder in Raum 2 gehen, sondern ihn Raum 1 betreten läßt.

```
501 ... 1,1,1,2,0,0
502 ... 2,1,1,3,0,0
```

Starten Sie das Programm, und lassen Sie sich von der Wirkung dieser kleinen Änderung überraschen.

Mit drei Räumen lassen sich auf diese Art und Weise bereits kleine Labyrinthe erstellen, wenn die Verbindungen wahllos, ohne Bezug zu einem Kompaß, ausgeführt werden, und nur ein einziger Zu- wie auch Abgang existieren.

Die übliche, bislang auch von uns verwandte Verbindungstechnik hält sich an die realen, durch Himmelsrichtungen vorgegebenen Wege. Dabei entsteht eine Welt, die eine problemlose Identifizierung der Räume und, besonders wenn der Spieler eine entsprechende Karte anfertigt, Orientierung gestattet.

Kleine Sprünge über einige Räume hinweg machen dieses Konzept jedoch schnell unbrauchbar.
Stellen wir uns drei hintereinander liegende Räume vor, die alle mit der Beschreibung ´Ich bin in einer großen Höhle´ aufwarten.
Ein Spieler, der von Westen in Höhle 1 gelangt, wird, da er nichts besonderes sieht, zunächst mit großen Schritten die nähere Umgebung kennenlernen wollen und vermutlich seine einmal eingeschlagene Richtung nicht ändern, da es ihm ein Bedürfnis ist, möglichst einfach den Weg zurückzufinden.

Nach Durchqueren von Raum 2 und 3 gelangt er wiederum in den ersten Raum, was er ohne entsprechende Handlungen aber unmöglich feststellen kann.

Durch Verwendung dieser Spungtechnik ist es möglich, ohne großen Aufwand riesig erscheinende Labyrinthe zu erstellen. Sechs Räume sind völlig ausreichend, dem Spieler das Finden des richtigen Weges fast unmöglich zu machen.
Selbst der Programmierer wird während der Testphase auf ausreichende Schwierigkeiten treffen !

Denn selbst die Anfertigung einer Karte wird bei dieser Arbeitsweise zum Problem, nicht jedoch bei einem alternativem Verfahren, das an dieser Stelle auch kurz vorgestellt werden soll.

Eine Vorgehensweise nach dieser Methode sieht zunächst ein maßstabsgerecht auf vorzugsweise kariertem Papier erstelltes Labyrinth vor, wie Sie es häufig auch in Rätselecken der verschiendensten Zeitschriften finden.
Bei der anschließenden Programmierung entspricht dabei jedes Kästchen einem Raum, wodurch der beträchtliche Aufwand bereits abzusehen ist.
Ebenso leicht ist es einzusehen, daß der Spieler während seiner Exkursion mit jedem neuen Raum ein neues Kästchen nimmt, die Wände markiert und die Durchgänge blank läßt.

Nehmen Sie stattdessen sechs Räume, von denen Sie jeweils einen als Eingangs-, als Sammel- und als Ausgangsraum festlegen. Dabei ist es für den Eingangsraum wichtig, daß er allein und auch nur in einer einzigen Richtung wieder zurück an den zuletzt besuchten Ort führt.
Entsprechendes gilt für den letzten Raum, nur ein einziger Durchgang erlaubt das Betreten von Neuland, alle anderen Wege müssen ins Labyrinth zurückführen.
Dem Sammelraum kommt dabei eine zentrale Funktion zu. Von jedem Raum des Labyrinthes führen zwei oder mehr Wege in diesen Raum, alle Ausgänge bringen den Spieler jedoch wieder in den ersten oder auch zweiten Raum des Irrgartens.

Um die Sache noch zu komplizieren, werden die Ausgänge des Raumes, der verbindungsmäßig vor dem letzten Raum liegt,

noch in der Art angelegt, daß nur ein einziger Weg in den letzten, alle anderen aber in den Sammel- oder in den ersten Raum führen.
Es erübrigt sich zu sagen, daß für alle Räume eine identische Beschreibung vorgesehen wird und daß möglichst alle sechs Richtungen benutzt werden.

Der Spieler wird nun fast soviel Glück wie ein Lottosieger benötigen, um das Labyrinth verlassen zu können: ehe er im vorletzten Raum aus sechs Wegen den einzig richtigen findet, gleiches gilt für den letzten Raum. Dabei wirft ihn jeder Fehler wieder in einen anderen Raum zurück, welcher wiederum sechs Möglichkeiten bietet.

Versuchen Sie, sich im folgenden Labyrinth zurechtzufinden - obwohl Sie selbst die Verbindungen erst kurze Zeit zuvor eingegeben haben, wird es Ihnen schwer fallen:

```
110 ar=14

507 DATA "am Eingangsstollen",7,0,1,5,0,
0
519 DATA auf einem kurvenreichen Gang,8,
0,0,5,0,0
524 DATA auf einem kurvenreichen Gang,10
,7,8,8,8,8
525 DATA auf einem kurvenreichen Gang,11
,8,8,13,11,8
526 DATA auf einem kurvenreichen Gang,11
,8,10,11,11,8
527 DATA auf einem kurvenreichen Gang,9,
8,10,11,10,8
528 DATA auf einem kurvenreichen Gang,8,
10,10,11,0,0
529 DATA auf einem kurvenreichen Gang,0,
11,12,11,0,0
```

ORTSWECHSEL

Außer Irrgärten in beliebiger Größe stehen uns weitere Mittel zur Verfügung, um dem Spieler die Orientierung zu erschweren.

Sehr großer Beliebtheit erfreuen sich dabei plötzliche Transfers der Hauptperson in einen anderen Raum, was dem Abenteurer durch Mitteilungen wie ´Alles um mich herum dreht sich´ deutlich gemacht wird.
Doch ist auch diese Nachricht bestenfalls als freundliche Geste des Programmierers aufzufassen, und nicht als Pflicht zu verstehen.

Auslöser der Aktionen solcher Art sind meist magische Gegenstände, zu deren Bedienung seit Aladin´s Wunderlampe ein kräftiges Reiben erforderlich ist.
Aber auch Zaubersprüche und -tränke verfehlen ihre Wirkung nicht, weshalb dem Spieler eines Adventureprogrammes nur geraten werden kann, allergrößte Vorsicht walten zu lassen, wenn in irgendeinem Zusammenhang das Wort ´Magie´ fallen sollte.

Programmtechnisch werfen diese Ortswechsel bei Verwendung unseres Adventure - Systems keinerlei Schwierigkeiten auf, denn es reicht aus, der Variablen SP die Nummer des neuen Raumes zuzuweisen.

´Reibe Lampe´ könnte daher folgendermaßen implementiert werden (Lampe = Objekt 3):

```
xxxx IF o=3 AND ob(3)=-1 THEN PRINT"Die
Welt dreht sich ...":spieler=9:GOTO 1080
```

Wenn dann auch noch die Raumbeschreibung von 9 der des alten Raumes entspricht, dürfte die Verwirrung des Spielers nach dem nächsten Zug unausweichlich bleiben. Dies gilt

besonders für den Fall, wenn die Meldung fortfällt oder ein anderer Text gewählt wird:

```
xxxx IF o=3 AND ob(3)=-1 THEN PRINT ok$;
" - nichts passiert.":spieler=9:GOTO 108
0
```

Eine weitere Steigerung des Schwierigkeitsgrades bedeutet es, wenn der Ortswechsel des Spielers nicht einmal mehr von den Manipulationen irgendwelcher Gegenstände abhängig ist, sondern einzig und allein vom Zufall gesteuert wird.

Wir können mit der Konstruktion unseres Adventureprogrammes vorschreiben, daß das Durchqueren eines bestimmten Raumes unumgänglich ist.
Entspricht der Inhalt von SPIELER der Nummer dieses Raumes, wird eine zufällige Entscheidung darüber getroffen, ob der Spieler versetzt wird oder ob ihm zwecks Ausführung weiterer Handlungen ein Aufenthalt gestattet wird.

Welcher Spieler, der nach erfolgter Durchquerung eines Irrgartens und anschließendem Betreten eines scheinbar harmlosen Raumes mehr als einmal wieder in das Labyrinth zurückgeworfen wurde, wird das Risiko einer Wiederholung dieses Vorganges eingehen ?

Wieder zeigt ein Ausschnitt aus dem Adventure Goldrausch, wie dieses Handicap eingebaut werden kann:

```
110 ar=16

519 DATA auf einem kurvenreichen Gang,8,
0,15,5,0,0
521 DATA in eimen Stollen,11,11,11,7,11,
0
522 DATA in einem Felsendom,0,15,0,0,0,0

1340 IF spieler=15 THEN IF RND(1)>.7 THE
```

N spieler=16

Versetzt in den kritischen Raum wird der Spieler durch Verlassen des Vorraumes zum Labyrinth, zur Zeit Raum acht, ein kurvenreicher Gang, in westlicher Richtung.

Aus diesem Raum kann er sich sofort wieder zurückziehen, oder er kann versuchen, herauszufinden, was die weitere Umgebung bringt.

Wenn der Zufall es will - und ob er will, legen wir mit dem Vergleichswert in der RND - Funktion fest -, gelangt er in den nächsten Raum (16), wenn nicht, findet er sich in einem Raum des Labyrinthes (Sammelraum 11) wieder, welcher ihm keine Anhaltspunkte zur Orientierung bietet.

War ihm das Glück nicht hold, wird er nach einigen Versuchen die Entscheidung treffen, daß es sich um einen weiteren Zugang des Irrgartens handelt, diesen Raum fortan meiden und somit einen Teil des Schatzes niemals finden.

VERSTECKTE ZUGÄNGE

können dem Spieler mindestens ebenso große Schwierigkeiten bereiten. Selbst wenn in der Zeile 'Ich kann nach ...' nur die Richtungen Osten und Westen genannt werden, muß deshalb die Welt im Norden nicht zu Ende sein.

Viele Spieler, soweit es sich um Anfänger handelt sogar die meisten, werden sich auf die gemachten Angaben verlassen und die Höhle niemals betreten. Denn dadurch, daß eine Untersuchung derselben, wie auch andere Aktionen, von der augenblicklichen Position möglich sind, wird jeder Verdacht auf eine großangelegte Höhlenwelt sofort zerstreut.

Erst eine Eingabe wie 'Betrete Höhle' führt zu neuen Erkenntnissen:

```
130 av=7

209 DATA betrete

2200 ON verbnummer GOTO 5000,6000,7000,8
000,9000,10000,13000

13010 IF o=13 AND spieler=5 THEN spieler
=12:PRINT ok$:GOTO 1080
```

ACHTUNG: Bevor Sie sich von der Funktion dieser Zeilen überzeugen, ist es erforderlich alle weiteren Räume einzugeben, wie auch die Verbindungen der bisher vorhandenen zu ändern (Zeilen 500 - 700).
Ebenso muß Zeile 110 an die neue Situation angepaßt werden.

Das schöne an dieser Realisation ist, daß der Zugang nicht einmal getarnt wird, im Gegensatz zu einer anderen Art der Ausführung, welche sich gleichfalls, zumindest seitens der Adventureproduzenten, großer Beliebtheit erfreut.

Aufgabe des Spielers ist es nun nicht mehr, nur einen versteckten Raum aufzuspüren und diesen mittels richtig gewählter Worte zu betreten, sondern er muß in der Lage sein, einen Plan zu entwickeln, wie ein begehbarer Weg dorthin konstruiert werden kann.

Im einfachsten Fall handelt es sich um eine Tür, die zunächst verschlossen ist. Eine entsprechende Anzeige durch den Treiber wird sowohl durch die Objektbeschreibungen wie auch die Daten der für diesen Raum geltenden Zeile der Richtungstabelle gewährleistet:

```
xxxx DATA eine Tuer,tuer,1

     REM RAUM 1
xxxx DATA in einer Zelle,0,0,0,0,0,0
```

Dieses übliche Verfahren sieht nun vor, daß der Spieler bei einer Untersuchung der Zelle die Tür, mit einem massivem Schloß versehen, entdeckt.

´Oeffne Tuer´ bzw. ´Oeffne Schloß´ weisen auf den fehlenden Schlüssel hin, wodurch der Spieler sich veranlaßt fühlt, alle weiteren Einrichtungsgegenstände des Raumes auf ihre Tauglichkeit zur Öffnung des Schlosses hin zu überprüfen.

Bei den meisten Adventures reicht die bloße Anwesenheit des Schlüssels im Inventory aus, die übrigen verlangen auch noch eine zweckentsprechende Benutzung des Schlüssels, eine Eigenschaft, die wir uns nun dank unserer Praxis erklären können.

Während die einen sich mit einem einfachen Test begnügen (ob(o)=-1), arbeiten die anderen, aufwendigeren Programme zusätzlich mit Flags.

Wie auch immer, steht die Tür erst offen, taucht eine entsprechende Richtungsangabe innerhalb der Kopfzeile auf, worauf wir auf eine Manipulation der für diesen Raum zuständigen Zeile des Feldes DURCHGANG (,) schließen.

```
xxxx IF o=y AND ob(z)=-1 AND spieler=r T
HEN PRINT ok$:durchgang(r,ri)=nr:GOTO 10
80

y = Objektnummer Tür
z = Objektnummer Schlüssel
r = augenblicklicher Raum
ri= betreffende Himmelsrichtung
nr= neu zu erreichender Raum
```

Obige Zeile, eingesetzt in den für die Behandlung des Verbes ´Oeffne´ vorgesehenen Programmteil, vergewissert sich davon, daß der Spieler im Besitz des richtigen Schlüssels (z) ist und vor der richtigen Tür steht (r). Anschließend werden die Angaben in der Richtungstabelle entsprechend geändert.

LIMITIERUNGEN UND ZÄHLER

halte ich für besonders geeignet, um ein Adventure aus der Masse der übrigen hervorzuheben. Vermitteln doch besonders sie jedem Abenteuerprogramm die Faszination der Realität, und gestalten sie einige Abschnitte des Adventures zu wahren Denksportaufgaben.

Ihr Aufgabenbereich ist dabei weniger in einer aktiven Spielbeteiligung zu sehen, als vielmehr in einer

Hintergrundkontrolle der Rahmenbedingungen, in die das Spiel eingebettet wird.

An erster Stelle muß hier wohl die Einschränkung des Inventories genannt werden, da ein Gewichtslimit inzwischen zum guten Ton dieser Computerspiele gehört.
Verständlich - erstens wirkt es unglaubhaft, wenn ein einzelner Mensch, wir gehen davon aus, daß es sich nicht um ein Supermann - Adventure handelt, beliebig viele Objekte mit sich herumschleppen kann, und zweitens kann das Spiel auch zu einfach werden, wenn stets alle Handlungsgegenstände sogleich zur Verfügung stehen.

Technisch gesehen überwacht ein Zähler die Anzahl der Gepäckstücke, die der Spieler mit sich führt, und ist die vom Programmierer vorgesehene Tragkraft der Hauptperson ausgelastet, wird die Aufnahme weiterer Gegenstände verweigert.

Es muß also vor Beginn aller Handlungen die maximal erlaubte Stückzahl festgelegt werden:

```
185 imax=5
```

Die Inventory - Routine lassen wir ansonsten wie sie ist, denn es ist der Vorgang des Nehmens, dem wir erhöhte Aufmerksamkeit schenken müssen.
Allen Aktionen mit dem Verb ´Nimm´ wird nun eine weitere gemeinsame Voraussetzung zugrunde gelegt, weshalb sich zum zweiten Male eine Änderung der Zeile 6000 anbietet.
Leider sieht diese Planung nicht nur eine Umstrukturierung des betreffenden Blockes vor, sondern sie macht auch unseren Traum eines universell einsetzbaren Treibers mit allen Funktionen und Restriktionen zunichte.

Aus diesem Grunde ändern wir die Sprungtabelle in Zeile 2200 und führen den erforderlichen Test vor Beginn der eigentlichen Befehlsausführung durch:

```
2200 ON verbnummer GOTO 5000,2210,7000,8
000,9000,10000
2201 REM ********** TEST OB PLATZ IM INV
2210 anzahl=0
2220 FOR i=1 TO ao
2230 IF ob(I)=-1 THEN anzahl=anzahl+1
2240 IF anzahl=imax THEN PRINT"Nein Dank
e. - Ich trage schon genug !":GOTO 1080
2250 NEXT i
2260 GOTO 6000 : REM   AUSFUEHRUNG NIMM
2270 REM ****************** ENDE INVTEST
```

Während diesem Zähler eine passive Rolle zukommt, und die Routine nur im Rahmen einer genau definierten Aktion aufgerufen wird, beschneiden andere Hintergrundkontrollen den Spieler nicht nur in der Freiheit seiner Handlungen, sondern können sogar zum Spielende führen.

Diese Behauptung gilt insbesondere für die folgenden Programmzeilen, deren Einbau in unser System dem Tüpfelchen auf dem ´i´ entspricht.

AVAILABLE LIGHT

ist die gebräuchliche Bezeichnung für eine bei einigen Adventureprogrammen anzutreffende Eigenart, die mit zusätzlichen Schwierigkeiten die Spieler zur Verzweiflung treiben kann.

Für Adventurespieler aus Passion scheinen die üblichen Hindernisse nicht ausreichend zu sein, und da bekanntermaßen nichts schwierieger ist, als das Leben in der Alltagswelt, muß zur Beseitigung dieses untragbaren Zustandes nichts weiter getan werden, als eben diese Welt noch perfekter nachzuahmen.

Ohne ´verfügbares Licht´ stehen wir im Dunkeln und werden nur schwerlich etwas sehen, geschweige denn präzise handeln können. Kommt dann noch die Tatsache hinzu, daß wir uns nicht in unserem Schlafgemach, wo dieser Zustand weitaus weniger hinderlich wäre, sondern in einer naturbelassenen, wilden und zerklüfteten Felsenlandschaft befinden, wird jeder weitere Schritt zu einem tödlichen Risiko.

Das kommt uns Adventureproduzenten natürlich sehr gelegen, und beim Einbau der nächsten Zeilen in unsere Adventures bereitet uns die Vorstellung der Reaktionen unserer Freunde, denen wir unser Werk selbstverständlich sofort nach Fertigstellung präsentieren wollen, eine zusätzliche Freude.

Glücklicherweise eignet sich jedes Abenteuerprogramm für einen Einbau dieses Hindernisses. Oft stehen weiträumige Höhlen oder unterirdische Gangsysteme zur Verfügung, andere Programme spielen sich in einem Haus ab; alles Orte, die einer künstlichen Beleuchtung bedürfen. Sollte tatsächlich einmal die Situation eintreten, daß die ganze Handlung unter freiem Himmel abläuft, muß eben der Lauf der Sonne nachvollzogen werden.

Um einen Sonnenauf- und -untergang realisieren zu können, reicht der Einsatz eines weiteren Zählers.

Zunächst legen wir fest, wie lange der Tag (die Nacht) dauern soll, wobei wir uns der Eingaben des Spielers als Zeiteinheit bedienen.
Nehmen wir an, 50 Züge nach Sonnenaufgang beginnt jeweils die Nacht. Dieser Wert wird ebenso typisch für jedes einzelne Adventure sein, wie die Anzahl der erreichbaren Punkte, weshalb die entsprechende Variable LM (Lichtmenge) gleichfalls bei Programmstart initialisiert wird:

```
186 lm=50:licht=-1
```

Ihr nun schon recht umfangreiches Wissen zum Thema Adventure erlaubt es Ihnen, die Variable LICHT als Signalschalter zu interpretieren, und zwar in dem Sinne, daß eine -1 eine für alle Handlungen ausreichende Lichtmenge, 0 aber absolute Dunkelheit bedeutet.

Die beiden möglichen Stellungen dieses Schalters zeigen der Ausgaberoutine unseres Treibers, ob der Spieler etwas sehen kann, oder ob er im Dunkeln tappen muß:

```
1131 REM ******************** KEIN LICHT
1132 IF licht THEN 1140
1133 PRINT"Ich weiss nicht genau, wo ich
bin."
1134 PRINT"Es ist zu dunkel, um etwas zu
sehen."
1135 PRINT: PRINT"Auch die Ausgaenge seh
e ich nicht mehr.":GOTO 1330
```

Nach Zeile 1132 werden die bislang benutzten Ausgaberoutinen abgearbeitet werden, falls Licht vorhanden ist.

Trifft das Gegenteil zu, informieren die Zeilen 1133 bis 1135 auf die gewohnte Art und Weise darüber, daß es nichts zu sehen gibt.
Mit dem Drucken der Trennungslinie kann der Programmverlauf durch die üblichen Zeilen fortgesetzt werden.

Ansonsten müssen wir nur noch für die jeweils richtige Stellung des Schalters LICHT sorgen, was nach Ablauf der mit LM festgelegten Anzahl von Spielzügen zu geschehen hat.

Der Zugzähler kann zu dieser Vergleichoperation nicht herangezogen werden, weil mit einem Tag- Nachtwechsel auch ein Rücksetzen des Zählers auf Null erforderlich wird; führen wir daher die Variable Lichtwechsel (LW) ein:

```
186 lm=50 : licht=-1 : lw=0

1080 zug=zug+1 : lw=lw+1
1084 IF lw=lm THEN GOSUB 3000


2999 REM *** UNTERPROGRAMM LICHTSCHALTER
3000 IF licht=-1 THEN licht=0:GOTO 3020
3010 IF licht=0 THEN licht=-1
3020 lw=0
3030 RETURN
```

Zu Beginn einer neuen Eingaberunde wird überprüft, ob die vorgesehene Zeitdauer (LM) einer Periode erreicht ist. Wenn ja, wird das Unterprogramm Schalter aufgerufen und der Zustand von LICHT gewechselt. Mehr ist nicht erforderlich, den Abenteurer einem Tag und Nachtwechsel auszusetzen, was die Suche nach einer Lampe erforderlich machen kann.

Für Goldrausch wollen wir uns jedoch nicht mit der Simulation des Sonnenlaufes begnügen, sondern machen das Geschehen vom Besitz einer Lampe abhängig.

Dadurch stellen wir dem Spieler die zusätzliche Aufgabe, immer genügend Brennstoff für die Lichtquelle bereitzuhalten.
In unserem Falle wird es sich dabei um Öl handeln, welches er in der Höhle des Bären finden kann. Dort füllt er es in die Flasche, die er zuvor wieder an sich genommen haben muß, und anschließend in die Lampe.
Jedes Füllen setzt dabei den Zähler Lichtwechsel wieder auf den Maximalwert.

Um die Sache zusätzlich zu komplizieren, wird die Laterne, zu dem Zeitpunkt, zu dem der Spieler sie findet, noch einen Rest Öl enthalten. Betritt er dann, ohne die Lampe nachzufüllen, die Tiefen der Mine, ist er unrettbar verloren. Ein weiterer Schritt, und der Abenteurer stolpert und bricht sich das Genick.

Da wir für das restliche Territorium keine Dunkelheit vorgesehen haben, besteht der erste Schritt zur Verwirklichung dieser Version in einer Änderung der Startwerte.
Eine Umschaltung erfolgt immer nur dann, wenn der Zähler Lichtwechsel gleich dem Inhalt von Lichtmenge ist.

Um dem Spieler zunächst jedoch eine fast unbegrenzte Anzahl von Spielzügen zu erlauben, legen wir LM mit Null und LW mit eins fest, und stellen dadurch sicher, daß LW nie gleich LM sein kann:

```
186 lm=0 : licht=-1 : lw=1 : l1=20
```

L1 kommt hinzu, weil die noch vorhandene Füllung geringer sein soll, als die später erfolgenden Nachfüllungen. Die übrigen Zeilen können vom vorangegangenen Beispiel übernommen werden.

Aktiviert wird diese Routine erst mit dem ersten Zünden der Lampe. Die Ausführung dieses Befehles wird neben der Ausgabe von ´O.K.´ auch die Zähler korrekt initialisieren:

```
11030 IF o=23 AND ob(23)=-1 THEN PRINT"O
.K. - Die Lampe brennt.":lm=l1:lw=1:GOTO
 1080
```

Für die nächsten zwanzig Aktionen steht dem Spieler nun ausreichend Licht zur Verfügung. Mit dem Nachfüllen der Lampe kann er diesen Wert auf fünfzig vergrößern:

```
12010 IF o=23 AND ob(23)=-1 AND fl(6) TH
EN l1=50:lm=50:lw=1:PRINT ok$:GOTO 1080
```

Um ein Nachfüllen des Brennstoffvorrates sowohl bei brennender als auch bei gelöschter Flamme zu ermöglichen, weisen wir der Variablen L1, wie auch LM, die Anzahl der nun durchführbaren Spielzüge zu.

Für den Fall, daß der Spieler sich an die Oberfläche begibt oder aus anderen Gründen die Lampe löscht, müssen wir den augenblicklichen ´Tankinhalt´ retten:

```
210 DATA loesche
2200 ... 14000
14000 IF o=23 AND OB(o)=-1 THEN l1=lm-lw
:PRINT ok$:GOTO 1080
```

Selbstverständlich müssen diese Werte auch mit dem Abspeichern eines Spieles gerettet werden:

```
1625 PRINT#9,spieler:PRINT#9,l1:PRINT#9,
lm:PRINT#9,lw

1725 INPUT#9,spieler:INPUT#9,l1:INPUT#9,
lm:INPUT#9,lw
```

Damit haben wir sichergestellt, daß die Sache mit dem Licht auch wirklich ihren vorgezeichneten Verlauf nehmen muß.

Um dem Spieler aber die plötzliche, unangenehme Überraschung des unabwendbaren Endes zu ersparen, denn sollte er sich in den Tiefen der Erde befinden, während ihm das Öl ausgeht, wird er seine Haut nicht mehr retten können, ist eine Warnung fairerweise angebracht.

Folgende Zeile wird ihn kurz vor Brennschluß der Lampe von der drohenden Gefahr unterrichten:

```
1350 IF lm-lw<15 THEN PRINT"Das Licht de
r Lampe wird schwaecher."
```

Ich hoffe, die vorangegangenen Beispiele werden Ihnen bei der Verwirklichung Ihrer eigenen Ideen helfen.
So wird es Ihnen sicherlich nicht schwerfallen, in Ihren Programmen beispielsweise dafür Sorge zu tragen, daß der Spieler rechtzeitig Nahrung zu sich nehmen muß, damit er weiterhin Dinge in die Hand nehmen und Aktionen durchführen kann.

Ebenso können Sie mittels eines weiteren Zählers Reaktionen von der verstrichenen Zeit abhängig machen.

ZUFÄLLE

Erdbeben, die eine Gesteinslawine auslösen, plötzlich auftauchende Monster, die natürlichen Feinde der Hauptperson im Adventure oder auch sich im unpassendem Moment öffnende Falltüren sind weitere, allseits beliebte Methoden, um den Spieler vom rechten Wege abzubringen.

Fast immer wird das Auftreten dieser Ereignisse vom Zufall gesteuert und das sogar in doppelter Hinsicht.

Ein Ungeheuer kann einen bestimmten Raum beleben und dem Spieler den Kopf abreißen:

```
1360 IF spieler=10 THEN m$(0)="Diesmal h
atte ich keinen Honig fuer den Baeren.":
GOTO 4500
```

Genausogut kann die Falle aber nur manchmal zuschnappen und wird zu anderen Zeiten dem Spieler kein Haar krümmen:

```
1370 IF spieler=20 AND RND(1)>.8 THEN m$
(0)="Der Boden brach unter mir ein.":GOT
O 4500
```

Die dritte Version stellt eine Kombination der bereits vorgestellten Elemente dar.
Grundsätzlich wird man auch von einer Beweglichkeit der Feinde des Abenteurers ausgehen können, weshalb auch dem Monster ein gewisser Abschnitt der Adventurewelt als Spielwiese zur Verfügung gestellt werden sollte:

```
1380 IF (spieler=13 OR spieler=14 OR spi
eler=16) AND RND(1)>.8 THEN m$(0)="Wiede
r der Baer.":GOTO 4500
```

VOM TEXT- ZUM GRAFIKADVENTURE

Neu vorgestellte Adventureprogramme weisen neben den besprochenen Features meist auch eine hervorragende optische Präsentation auf. Für jeden Raum des Adventures existiert ein Bild, welches den Spieler mehr oder weniger gut über spielwichtige Dinge und deren Umfeld informiert.

Zur prorammtechnischen Ausführung dieser Grafikadventures stehen dem Programmierer prinzipiell drei Wege offen.

Zunächst kann er auf der Diskette für jeden Raum und jeden Gegenstand eine entsprechende Grafik speichern und diese bei Bedarf in den Bildschirmspeicher des Computers laden. Dieser Weg wird heutzutage bei fast allen Grafikadventures eingeschlagen, einerseits weil sich mittels Videodigitalisierer oder anderer Hilfsmittel, wie beispielsweise einem Grafiktablett großartige Ergebnisse erzielen lassen, andererseits besteht der zusätzliche Programmieraufwand nur aus einem einzigen Unterprogramm, welches abhängig von der Raumnummer (SPIELER) eine Datei laden muß, und daß im günstigsten Fall aus einer einzigen Zeile besteht.
Natürlich wird entsprechend viel Platz auf der Diskette benötigt (jetzt wissen Sie, warum manche Adventures über drei und mehr Disketten gehen), wie der Spieler auch die Ladezeit geduldig ertragen muß.

Die zweite Methode arbeitet mit einem Grafikinterpreter, einem Programmteil des Adventures, welcher über einen Line- und Paintbefehl verfügt, und dem jeweils die Daten aller Umrisse und Farben zur Verfügung gestellt werden müssen.
Dieses Verfahren geht am sparsamsten mit dem zur Verfügung stehendem Speicherplatz um, denn für jeden Punkt ist im allgemeinen nur ein Byte erforderlich, und wenn die Linien als Linienzüge ausgeführt werden, reicht zu ihrer Speicherung ebenfalls jeweils eine Speicherstelle.

Übrigens arbeitet der im folgenden vorgestellte Grafik - Programmer nach diesem Prinzip, wenn er auf Druck von Z ein Bild neu zeichnet.

Das dritte Verfahren schließlich besticht durch seine Schnelligkeit, denn jede für den Aufbau der Grafik notwendige Anweisung wird direkt ausgeführt. Voraussetzung ist natürlich, daß das Betriebssystem des Computers entsprechende Befehle zur Verfügung stellt und daß der Arbeitspeicher die erforderlichen Anweisungen aufnehmen kann.

Bei der weiteren Entwicklung unserer Adventures wollen wir uns auf die letzten beiden Methoden beschränken, denn zu deren Anwendung bedürfen wir weder technischer Hilfsmittel noch aufwendiger Unterprogramme.

Beide Lösungen erfordern natürlich einige kleine Änderungen an unserem Treiberprogramm, denn die Ausgaben werden nun an anderen Stellen des Bildschirmes gemacht werden müssen.

Unser nächster Schritt besteht nun darin, alle Bildschirmausgaben derart umzudirigieren, daß nur noch einige der untersten Zeilen benutzt werden.
Glücklicherweise kommmen uns auch hier die hervorragenden Eigenschaften des Locomotive Basic entgegen, so daß wir uns umständliche Änderungen unseres Treibers ersparen können.

Zwei Fenster, und zwar Window 0 für die üblichen Ausgaben und Window 1 für den Grafikbereich, sind völlig ausreichend; es ist nicht notwendig, sämtliche Print - Befehle zu ändern, da #0 dem Standardstream entspricht und somit alle Ausgaben umdirigiert werden.

```
1050 WINDOW#1,1,40,1,16
1070 WINDOW#0,1,40,17,25:PAPER 2:INK 2,1
4:CLS#0
```

DER BILDAUFBAU

Den Aufbau der jeweiligen Grafik überlassen Sie einigen Unterprogrammen, die vom Treiber aus aufgerufen werden. Allerdings gilt es zu beachten, daß diese Druckroutine nicht nach jeder Eingabe des Spielers aufgerufen werden muß, denn warum sollte nach jeder Aktion innerhalb eines Raumes dieser neu gezeichnet werden ?

Führen wir daher eine weitere Kontrollvariable ein:

```
1080 PRINT:IF alt<>spieler THEN GOSUB
30000
1140 PRINT"Ich bin ";:alt=spieler
```

Muß ein Raum auf dem Bildschirm dargestellt werden, so wird der Block mit den Grafikroutinen angesprungen. Anschließend wird dessen Nummer an die Variable ALT übergeben. Entfernt der Spieler sich dann vom Ort des Geschehens, so ist die Bedingung in Zeile 1080 erfüllt und es wird wiederum Zeile 30000 aufgerufen, welche anhand der Raumnummer die richtige Zeichenroutine wählt:

```
30000 ON spieler GOSUB 31000,32000,330
00
30010 RETURN

31000 REM RAUM 1 ZEICHNEN
31999 RETURN

32000 REM RAUM 2 ZEICHNEN
31999 RETURN
```

Anschließend wird der Programmlauf mit der Ausgabe der Objektbeschreibungen in gewohnter Weise fortgesetzt.

Sollten Sie es nun vorziehen, den Aufbau der Grafiken mittels eines Interpreters vorzunehmen, so bauen Sie ab Zeile 30000 die entsprechenden Routinen des später vorgestellten Grafik - Programmers ein.
Sie benötigen nur die zwei Module Bild laden und Bild zeichnen, welche bei Programmzeile 7000 beziehungsweise bei Zeile 5000 beginnen.

5. KAPITEL
- DAS SYSTEM -

Die vorangegangenen Kapitel haben Ihnen das nötige Rüstzeug in die Hand gegeben, um gute Adventures zu schreiben.
Dennoch, obwohl wir mit unserem System ein Konzept entwickelt haben, welches zur Entwicklung der verschiedensten Adventures nur die Programierung der Daten, Bedingungen und Aktionen des betreffenden Spieles erforderlich macht, ist weiterhin ein beträchtlicher Arbeitsaufwand vonnöten, bevor die Testphase mit dem ersten Spielversuch beginnen kann.

Eine Vereinfachung der Arbeit bedeutet ein sogenannter Adventure - Generator, der, nachdem alle notwendigen Daten eingegeben wurden, auf der Diskette beziehungsweise Cassette ein fertiges, spielbereites Programm hinterläßt.

Nach diesem Verfahren arbeitet ´VENTUREFIX´, ein Programmgenerator welcher lauffertige Basic-Programme erzeugt und dessen Listing Sie im nächten Kapitel finden werden.
Diese Lösung erfordert allerdings immer noch, daß die Planung des Spieles vor Beginn der Realisation vollständig abgeschlossen ist. Denn bevor der Generator mit der Arbeit beginnen kann, hat der Adventureproduzent eine Liste aller Räume, Objekte, Verben usw. zu erstellen und diese Daten dann in einer Mammutsitzung einzugeben.

Treten während eines Testlaufes des so erstellten Programmes Fehler hervor, müssen diese weiterhin auf die übliche Art beseitigt werden.

Weitaus wünschenswerter wäre daher eine Lösung, die jeden einzelnen Gedanken des Autors sofort fixiert und darüberhinaus eine sofortige Überprüfung und auch Besserung ermöglicht.
Dies kann jedoch nur geschehen, wenn nicht ein fertiges Programm erzeugt, sondern eine Datei entwickelt wird, die alle für einen Spielablauf notwendigen Daten enthält.

Die Auswertung dieser Daten fällt dann einem Interpreter zu, der die zur Ausführung der verschiedensten Aktionen notwendigen Routinen enthält.

Wie solch ein Interpreter und der dazugehörige Editor auszusehen haben, wollen wir Ihnen in diesem Kapitel des Buches zeigen.

DIE DATEI

Es wurde bereits deutlich gemacht, daß das eigentliche Adventure nunmehr nur noch durch eine Datei repräsentiert wird.
Diese Datei muß alle für ein bestimmtes Spiel notwendigen Angaben in einer genau definierten Anordnung enthalten, damit für den Interpreter die Voraussetzung geschaffen ist, diverse Adventures verschiedener Länge zu laden.

Nehmen wir eines unserer programmmäßig realisierten Adventures unter die Lupe, so zeigt sich, daß diese bereits zu einem großen Teil interpretierend arbeiten.

Die Verben, wie auch die Objekte, werden nicht direkt zur Ausführung einer Aktion herangezogen, sondern es werden zunächst zwei Nummern ermittelt, deren Kombination genau festlegt, welche Eingabe der Spieler gemacht hat.

Als Voraussetzung für diese Arbeitsweise war es erforderlich, die Worte in Variablen einzulesen, ein Vorgang, der statt der Data - Zeilen auch direkt eine Datei von einer Diskette (Kassette) benutzen kann.

Auch die spieltypischen Daten haben wir bereits ermittelt und in den Zeilen 100 bis 185 als Kenndaten des Adventures in das Programm eingebracht.

Überlegen wir uns nun, welche Werte aus diesem Block für einen Interpreter notwendig sind, der zwar nicht alle Extras unseres System beinhaltet, aber eine ausreichende Anzahl von Funktionen bietet, um ein Spiel möglich zu machen.

Zunächst finden wir einige Informationen über die Anzahl der vorhandenen Räume, Gegenstände und Verben.
Diese Grenzwerte erwiesen sich zur korrekten Steuerung der einlesenden Schleifen als notwendig und werden es aus gleichem Grunde auch für den Interpreter sein.
Ebenso werden wir die genaue Anzahl aller Mitteilungen festlegen müssen, denn sie werden fortan ohne Ausnahme in einem Variablenfeld aufbewahrt, und beizeiten ausgedruckt werden.

Gleiches gilt für die Bedingungen und Aktionen. In einem Feld, ähnlich der Richtungstabelle DURCHGANG(,) werden wir diese Spieldaten zum Abruf bereithalten, weshalb wir die Variablen AC und BC (Aktionscodes und Bedingungscodes) zusätzlich in die Liste der Kenndaten aufnehmen.

Auf die Anzahl der Flags dürfen wir ebenfalls nicht verzichten, denn auch unserem Interpreter soll ein Abspeichern des Spielstandes möglich sein.

Als weiteren, für das Spiel wichtigen Wert müssen wir noch die Startposition des Spielers in unsere Datei aufnehmen.
Die Funktion des Systems wäre zwar auch gewährleistet, wenn SPIELER nicht bei Spielstart initialisiert werden würde, aber dann müßte jedes Adventure immer in Raum 1 beginnen.
Auf den ersten Blick wird daraus zwar kein Nachteil ersichtlich, aber stellen Sie sich vor, nach einer kleinen Umstellung der Handlung soll der Abenteurer an einem völlig anderen Ort mit dem Spiel beginnen.

Findet sich dann keine Variable SPIELER, die entsprechend neu initialisiert werden kann, müssen Sie alle Raumbeschreibungen, der neuen Planung gemäß umschreiben.

Auf eine Punktwertung können wir zunächst verzichten. Damit aber das Ende des Spieles definiert werden kann, werden wir eine entsprechende Aktion vorsehen, die beispielsweise die Flags oder die im Inventory befindlichen Objekte auswerten kann.

Eine Ergänzung, die nichts mit dem Spielablauf zu tun hat, aber dennoch sinnvoll ist, sollten wir ebenfalls nicht vergessen.
So empfiehlt es sich, in der Datei auch Informationen über den Namen des Autors, den Namen des Adventures und vor allem das Datum der letzten Bearbeitung bereitzusellen, damit Sie die Version 5 mit nur noch drei Fehlern von der Version 1 mit mehr als zehn Fehlern, unterscheiden können.

DER AUFBAU DES EDITORS

Die Aufgabe des Editors ist es nun, dem Anwender die Eingabe wie auch später erforderliche Änderungen dieser Daten, auf bequeme und überschaubare Art möglich zu machen. Dazu werden eine Reihe unterschiedlicher Programmteile notwendig sein, deren Aufruf durch ein Menü erfolgen soll.

Die Reihenfolge wird dabei in gewissem Maße vorgegeben sein, denn solange die Räume noch nicht vorhanden sind, können die Gegenstände auch nicht plaziert werden.

Im einzelnen könnte der Arbeitsablauf der Konstruktion eines Adventures mit dem Editor folgendermaßen aussehen:

1. *RÄUME EINGEBEN*
2. *OBJEKTE EINGEBEN*
3. *VERBEN EINGEBEN*
4. *OBJEKTE PLAZIEREN*
5. *RÄUME VERBINDEN*
6. *MITTEILUNGSTEXTE EINGEBEN*
7. *BEDINGUNGEN & AKTIONEN EINGEBEN*

8. *FERTIGE DATEI SPEICHERN*

Jede gewünschte Funktion wählt der Anwender durch Eingabe der entsprechenden Ziffer, weshalb wir im Anschluß an die für obiges Bild erforderlichen Ausgabebefehle eine Programmzeile mit den Sprungzielen vorsehen:

```
350 ON a+1 GOTO 3000,1100,1200,1300,1400
,1500,4000,1600,5000,6000
```

DIE EINGABEN

Bevor wir mit dem Aufbau der einzelnen Unterprogramme beginnen, hilft uns folgende Überlegung, den zu treibenden Aufwand möglichst gering und den Editor kurz zu halten.

Der Benutzer muß jeweils über die erwarteten Daten informiert werden, damit er korrekte Eingaben machen kann. Zweckmäßigerweise konstruieren wir einfache Eingabemasken, die entsprechende Hinweise enthalten.
Diese Masken werden, zumindest bei Eingabe der Räume, der Gegenstände und der Verben, weitgehend identisch sein, weshalb sich ein gemeinsamer Programmteil für diese Aufgaben anbietet.

Damit dieses Unterprogramm die richtigen Anweisungen geben kann, müssen diese Titel vor dem Aufruf zur Verfügung gestellt werden:

```
1100 CLS):t1$="ORTE EINGEBEN ":t2$=" RAU
M NR.:":t3$="ICH BIN "
1105 z=ar
1110 GOSUB 510
```

Anschließend wird der Variablen Z die Anzahl der bisher vorhandenen Räume mitgeteilt und die Ein/Ausgabemaske aufgerufen (1110).

Zu ihrer Gestaltung muß der Cursor immer wieder an anderen Stellen positioniert werden:

```
499 REM ****** UNTERPROGRAMM EIN/AUSGABE
500 CLS
510 LOCATE 1,1:PRINT m4$:LOCATE 1,1:PRIN
T t1$
520 z=z+1
530 LOCATE 25,1:PRINT t2$;z
540 PRINT m3$
590 LOCATE 1,25:PRINT t3$;:eingabe$="":L
INE INPUT eingabe$
600 IF LEN(eingabe$)=0 THEN GOTO 200
610 IF a=2 THEN PRINT:LOCATE 1,25:INPUT"
Rufname ";rn$(z):PRINT
620 PRINT
640 RETURN
650 REM ****************** ENDE EIN/AUS
```

Zunächst werden die entsprechenden Hinweise ausgedruckt. In Zeile 610 wird überprüft, ob die gerade aufgerufene Funktion mit der Eingabe der Objekte identisch ist.
Zeile 590 nimmt die Eingabe entgegen, die Zuweisung an die richtige Variable erfolgt innerhalb des rufenden

Unterprogrammes (mit Ausnahme des zweiten Objektnamens, der in Zeile 610 direkt eingegeben wird).

Die Beendigung der aufgerufenen Funktion gilt als beabsichtigt, wenn statt einer Eingabe nur die ENTER - Taste gedrückt wird, die Länge des Eingabetextes somit gleich Null ist (Zeile 600).

Analog zu diesem Beispiel wird die Eingabe der Objekte und Verben vorgenommen. Erst die Verbindung der Räume erfordert einen anderen Aufbau unseres Editierprogrammes.

VERBINDUNGEN DER RÄUME

Zunächst müssen alle möglichen Räume angezeigt werden, damit der Anwender die richtigen Raumnummern eingeben kann. Da diese Funktion ebenfalls für die Plazierung der Objekte notwendig ist, erstellen wir ein weiteres Unterprogramm (12000 - 12070).

Daran anschließend werden für alle Räume der Reihe nach die sechs Richtungen abgefragt und die Eingaben direkt in die entsprechende Position der Richtungstabelle geschrieben:

```
1499 REM ************** RAEUME VERBINDEN
1500 FOR r1=i2 TO ar
1510 CLS
1515 PRINT
1520 GOSUB 12000
1530 PRINT "Raum";r1;"fuehrt im Norden i
n Raum";:INPUT du(r1,1)
1540 PRINT "Raum";r1;"fuehrt im Sueden i
n Raum";:INPUT du(r1,2)

...
```

```
1585 NEXT r1
1590 GOTO 200
1591 REM ************* ENDE VERBINDUNGEN
```

AKTIONEN & BEDINGUNGEN

Bevor wir mit der Eingabe der wichtigsten Daten, den Bedingungen und Aktionen fortfahren, müssen wir ein System zur Codierung entwickeln.

Zur Aufbewahrung der Daten verwenden wir ein weiteres zweidimensionales Feld, wobei die Reihen diesmal durch die Verben, und die Spalten durch die Objekte vorgegeben werden.
Um die Bedingungen auch für uns interpretierbar zu machen, entscheiden wir uns für eine Codierung durch Buchstaben. Zahlen würden bei der Bearbeitung zwar einen Geschwindigkeitsvorteil bedeuten, aber die Bedeutung einer zwölfstelligen Zahl zu erkennen, ist mit Sicherheit schwieriger, als die Interpretation eines Strings wie "RF3S22" vorzunehmen, wenn dabei Konventionen, wie aus den folgenden Zusammenstellungen ersichtlich, getroffen wurden.

R - OBJEKT UND SPIELER IM GLEICHEN RAUM
I - OBJEKT IST IM INVENTORY
N - OBJEKT IST NICHT IM RAUM
FX- FLAG X IST GESETZT
GX- FLAG X IST GELOESCHT
SXX SPIELER MUSS SEIN IN RAUM XX

Diese Bedingungen sind völlig ausreichend, um Aktionen des Spielers nur zu den von der Handlung vorgeschriebenen Zeitpunkten durchführbar zu machen.

Auch die vorgesehenen Aktionen entsprechen im wesentlichen den auf Seite 97 gemachten Anmerkungen, neu hinzugekommen sind lediglich die Befehle Dxy und E.
Der Befehl D soll den Interpreter veranlassen, einen Ausgang aus dem augenblicklichen Raum nach Raum x in Richtung y sichtbar werden zu lassen, E beendet das Spiel und informiert den Spieler über seinen Sieg.

```
V    - OBJEKT VERSCHWINDET
I    - GEGENSTAND KOMMT INS INVENTORY
NXX  - OBJEKT MIT NUMMER XX ERSCHEINT NEU
DXY  - DURCHGANG NACH RAUM X IN RICHTUNG Y
SXX  - SPIELER WIRD IN RAUM XX VERSETZT
FX   - FLAG X WIRD GESETZT
LX   - FLAG X WIRD GELOESCHT
MXX  - MELDUNG NR. XX WIRD AUSGEGEBEN
T    - SPIELFIGUR STIRBT
E    - SPIELENDE DURCH SIEG DES SPIELERS
```

DIE MITTEILUNGEN

können ohne weitere Umstände sofort bei der Eingabe in die richtigen Variablen geschrieben werden. Entsprechend kurz fällt der für diesen Zweck vorgesehene Programmteil aus:

```
3999 REM ****************** MITTEILUNGEN
4000 CLS:PRINT"Mitteilungen eingeben"
4010 PRINT m3$
4020 am=am+1
4030 PRINT am;:INPUT m$(am)
4040 IF LEN(m$(am))=0 THEN am=am-1:GOTO
200
4050 GOTO 4020
4060 REM ************* ENDE MITTEILUNGEN
```

AM behält die Kontrolle über die Anzahl der eingegeben Nachrichten und muß nach Beendigung der Funktion durch Eingabe eines Sternchens vermindert werden.

KASSETTEN- ODER DISKETTENZUGRIFFE

werden erforderlich, um die Datei zu speichern bzw. um sie zur erneuten Bearbeitung wieder in den Rechner zu laden.
Damit die Datei, auch wenn ein Spiel noch nicht vollständig realisiert wurde, für den Interpreter bereits spielbar ist, müssen vor einem Abspeicherungsvorgang zusätzliche Daten eingegeben werden.

Wichtig für den Interpreter ist vor allen Dingen die Wortlänge, denn dieser soll den für Variablen vorgesehenen Speicherplatz nicht dadurch mehr als nötig belegen, daß er die vollständigen Wörter speichert, sondern er soll nur die wirklich benötigten Buchstaben einladen.

Bereits erwähnt wurde, daß der Startraum des Spielers in der Datei enthalten sein muß, wie auch die Anzahl der verwendeten Flags für den Interpreter wichtig wird, sobald die SAVE GAME Funktion aufgerufen wird.

Nach Eingabe dieser Werte werden alle Daten als sequentelles File auf dem Datenträger abgelegt.

Ansonsten weist das Listing des Adventure - Editors keine Besonderheiten auf, so daß Sie, nicht zuletzt auch wegen der Remarks, die auf die einzelnen Blöcke hinweisen, die Funktionsweise verstehen dürften und das Programm nach Ihren Wünschen erweitern können.

DER ADVENTURE INTERPRETER

Auch das Verständnis des Interpreters wird Ihnen kaum Schwierigkeiten bereiten, ist er doch ähnlich unserer Adventureprogramme aus drei Blöcken aufgebaut.

Im ersten Teil des Programmes werden wieder die Variablen mit den für das Spiel erforderlichen Daten initialisiert. Statt der DATA - Zeilen mit zugehörigen READ - Anweisungen benutzen wir nun ein mit dem Editor erstelltes Datenfile.

Nachdem Sie die gewünschte Datei gewählt haben, wird sie von den Zeilen 60 bis 210 ausgewertet und aufbereitet.

Der Programmblock von Zeile 1000 bis einschließlich Zeile 2160 entspricht dem von uns entwickeltem Treiber, Änderungen sind nicht erforderlich.

Völlig neu ist jedoch die Art der Ausführung jeglicher Aktionen.
Ein einziges Programm, der Interpreter, soll in der Lage sein, zahlreiche, jeweils voneinander verschiedene Spiele ablaufen zu lassen. Deshalb ist es nicht möglich, starre Strukturen, die immer nur auf eine einzige Situation zugeschnitten sind, zu verwenden. Aus einer Reihe von Einzelaktionen müssen die gerade erforderlichen herausgesucht und ausgeführt werden, bis die Summe ihrer Wirkungen das gleiche Resultat erzielt hat.

Diese Lösungsstrategie verfolgen wir sowohl bei der Überprüfung der Bedingungen als auch zur Ausführung der Handlungen.

```
2200 FOR ab=1 TO LEN (bc$(vn,o))
2210 bd$(ab)=MID$(bc$(vn,o),ab,1)
2220 NEXT ab
```

Zunächst wird der Endwert der Schleife ermittelt, der von der Anzahl der für die Codierung einer bestimmten Aktion benutzten Buchstaben abhängig ist. Anschließend erfolgt eine Zerlegung des gesamten Strings in die einzelnen Bedingungen, die jeweils einem Element der Liste BD$() zugewiesen werden.

Eine weitere Schleife überprüft nun die Erfüllung jeder einzelnen Voraussetzung innerhalb einer weiteren Schleife.

Jedoch war es bei der Codierung erforderlich, für einige Bedingungen zwei und mehr Parameter festzulegen. So erfordert der Code S (Anwesenheit des Spielers in einem bestimmten Raum) die Angabe einer Raumnummer, wie auch das betreffende Flag bei F und G genannt werden muß.
Aus diesen Gründen verzichten wir auf eine For/Next - Schleife und wählen stattdessen einen Aufbau, der es uns gestattet, den Schleifenzähler um beliebige, differierende Werte zu erhöhen.

Neben dieser Variablen (X) benutzen wir weiterhin die Variablen ERGEBNIS und OK, wobei Ergebnis nach jedem Schleifendurchgang logisch wahr (-1) sein soll, wenn die Bedingung erfüllt wurde, und logisch falsch, wenn die zu überprüfende Teilbedingung nicht erfüllt ist.
Der Wert dieser Aktion wird sodann mit dem zuvor ermittelten Ergebnis durch 'und' logisch verknüpft.
Somit wird sichergestellt, daß eine Handlung nur durchführbar wird, wenn alle Bedingungen erfüllt sind (denn nur dann wird ok = -1 sein !).

```
2300 x=0:er=0:ok=-1
2310 x=x+1:ok=(ok AND er)
```

Jeder Schleifendurchgang beginnt mit einer Erhöhung des Zählers, erst dann wird geprüft, ob der Endwert, der ja von der Anzahl der Bedingungen abhängig ist, erreicht ist oder nicht.

```
2320 IF x=ab+1 THEN 2500 : REM ALLE BEDI
NGUNGEN UEBERPRUEFT
```

Abhängig von ok wird nun entschieden, ob eine Reaktion stattfindet, oder ob der Spieler weitere vorbereitende Handlungen durchführen muß:

```
2500 IF NOT ok THEN PRINT"Das geht im Mo
ment nicht.":GOTO 1080
3999 REM **** ALLE BEDINGUNGEN ERFUELLT
4000 PRINT "Okay !"
```

Innerhalb des Schleifenrumpfes werden die Bedingungen der Reihe nach überprüft, wobei die betreffende Programmzeile auf die gleiche Art ermittelt wird, wie beispielsweise die

Routinen zur Behandlung der Ein - Wort Befehle innerhalb unseres Treiberprogrammes:

```
2330 IF bd$(x)<>"R"THEN 2350
2340 IF ob(o)=spieler THEN er=-1
2345 GOTO 2310
2350 IF bd$(x)<>"I"THEN 2370
2360 IF ob(o)=-1 THEN er=-1
2365 GOTO 2310
2370 IF bd$(x)<>"N"THEN 2390
2380 IF (ob(o)<>spieler AND ob(o))<>-1 T
HEN er=-1
2385 GOTO 2310
2390 IF bd$(x)<>"S"THEN 2410
2400 r1$=bd$(x+1)+bd$(x+2):x=x+2:IF spie
ler=VAL(r1$) THEN er=-1
2405 GOTO 2310
2410 IF bd$(x)<>"F" THEN 2450
2420 IF fl(VAL(bd$(x+1)))=-1 THEN 2440
2430 x=x+1:GOTO 2450
2440 er=-1:x=x+1:GOTO 2310
2450 IF bd$(x)<>"G" THEN 2310
2460 IF NOT fl(VAL(bd$(x+1))) THEN 2480
2470 x=x+1:GOTO2310
2480 er=-1:x=x+1:GOTO 2310
2490 GOTO 2310
```

Außer den Zeilen 2400 bis 2470 werden auch diese Zeilen Sie vor keinerlei Probleme stellen.
In Zeile 2400 werden auch noch die nächsten zwei Elemente des Bedingungscodes BD$ herangezogen, um die Raumnummer zu ermitteln, anschließend wird der Zähler aktualisiert und eventuell die Ergebnisvariable auf wahr gesetzt.
Ähnlich arbeiten die für die Auswertung der Flags vorgesehenen Zeilen.

Identisch ist die Arbeitsweise der einen Befehl ausführenden Programmzeilen.
Innerhalb dieses Blockes wäre nur auf die Zeilen 5220 bis 5240 hinzuweisen, die es ermöglichen, beispielsweise eine Tür zu öffnen. Durch ihre Manipulationen werden die korrekten Werte in die Richtungstabelle geschrieben, so daß einerseits der neue Durchgang sichtbar wird (5220), andererseits aber auch ein Rückweg vorhanden ist, wenn der Spieler sich in den neuen Raum begeben hat (5230).

Die Zeilen von 5500 bis 5700 enthalten jeweils eine kurzes Unterprogramm, welche der Ermittlung einer zweistelligen Zahl (5500), einer einzigen Ziffer (5600) beziehungsweise zweier Ziffern dienen (5700).
Benötigt werden diese Werte zur Ausgabe der richtigen Mitteilungen, der korrekten Behandlung der Flags wie auch für die Öffnung eines Durchganges zu einem bestimmten in eine der sechs Richtungen.

Das vollständige Listing des Interpreters entnehmen Sie bitte ebenfalls dem nächsten Kapitel

6. KAPITEL

- ADVENTUREPRAXIS -

Dieses Kapitel wird Ihnen weniger zeigen, wie man´s macht, sondern zeigt Ihnen am Beispiel vollständiger Programme, was man machen kann.

Es folgen die kompletten Listings von drei Adventurespielen, die alle unter Verwendung der in diesem Buch entwickelten Programmteile und Routinen geschrieben wurden.

Zum einen handelt es sich dabei um *Goldrausch*, ein Adventure, zu dem schon fast zuviel gesagt wurde; als zweites Programm wäre das *Verzauberte Schloß* zu erwähnen. Die Krönung des Ganzen ist aber *Space Mission*, ein trickreiches Spiel, welches als Grafikadventure realisiert wurde.

Leider ist es jedoch so, daß Sie, wenn Sie dieses Buch durchgearbeitet haben, die Programme fast wie eine Anweisung zur Lösung lesen können. Deshalb wäre es empfehlenswert, die Arbeit des Eintippens mit jemandem zu teilen.

Dies gilt insbesondere für die Programmzeilen ab Nummer 5000, da hier das Spielgeschehen programmiert ist.

Im Anschluß an die Spiele finden Sie dann die versprochenen Programmgeneratoren, nämlich den Grafik - Programmer, ´Venturefix´, wie auch die Programme des Adventuresystems.

Diese Programme machen es Ihnen möglich, menügesteuert Adventures zu erzeugen und diese sogar mit einem Minimum an Arbeit zu eindrucksvollen Grafikprogrammen auszubauen.
Die notwendigen Erläuterungen und Bedienungshinweise stehen den betreffenden Programmlistings voran.

SPIELANLEITUNG: ADVENTURES

Nachdem Sie das betreffende Programm geladen und mit RUN gestartet haben, erscheinen nach dem Titelbild und weiteren, allgemeinen Spielhinweisen die ersten Beschreibungen und Mitteilungen auf dem Monitor.
Sie müssen zwischen der oberen und unteren Bildschirmhälfte unterscheiden: - in der oberen Hälfte finden Sie eine Beschreibung des Ortes, an dem Sie sich gerade befinden, daran anschließend werden Ihnen alle sichtbaren Gegenstände aufgezählt, und in einer weiteren Zeile werden sämtliche Richtungen genannt, in die Sie sich bewegen können.
Die untere Hälfte dient hingegen dazu, Ihre Befehle an die Spielfigur einzugeben, ebenso werden dort die von der Durchführung Ihrer Eingabe abhängigen Informationen an Sie ausgegeben werden.

IHRE EINGABEN:

Ihre üblichen Eingaben werden meist aus zwei Worten, einem Verb und einem Objekt, bestehen. Überprüfen Sie zunächst, welche Eingabelängen das vorliegende Adventure fordert, meist sind die ersten drei oder vier Buchstaben zur Identifikation eines Wortes ausreichend.
Achten Sie darauf, Bezeichnungen zu verwenden, wie sie auch innerhalb der Beschreibngen verwendet werden.
Das bedeutet allerdings nicht, daß Sie auf eine Mitteilung wie:

ICH SEHE EINE ROTE TÜR.

WAS SOLL ICH TUN:

mit OEFFNE ROTE TUER reagieren, meistens wird die Eingabe OEFFNE TUER völlig ausreichend sein.

Zusätzlich werden Ihnen einige weitere Eingaben erlaubt sein, die sogenannten Ein - Wort Befehle, welche weniger dem Spielverlauf dienlich sind, als daß sie Ihnen gewisse Serviceleistungen anbieten, um die Spielweise zu vereinfachen.

RÄUMLICHE FORTBEWEGUNG:

Sie können sich generell immer in die angegebenen Richtungen bewegen. Dazu reicht es aus, den Anfangsbuchstaben der jeweiligen Richtung einzugeben (außer für Bewegungen nach oben, hier müssen Sie zur Unterscheidung von Osten OB eingeben): Norden, Süden, Osten, Westen : N, S, O, W; Oben, Unten : OB, U.

Es gibt jedoch auch Ausnahmen und manchmal kann eine Eingabe wie BETRETE ... nutzbringend sein.

MANIPULATION VON OBJEKTEN:

Natürlich wollen Sie in der imaginären Welt nicht nur auf eine Besichtigungsreise gehen, sondern Sie müssen handeln und eine Aufgabe lösen.
Dies setzt voraus, daß Sie die Gegenstände dieser Welt zunächst erst einmal entdecken, anschließend können Sie sie UNTERSUCHEN, NEHMEN, BENUTZEN usw. (Wenn Sie einmal nicht mehr weiter wissen, sollten Sie sich an diese Worte erinnern !).
Um möglichst schnell ans Ziel zu kommen, sollten Sie bei Ihrem weiteren Vorgehen möglichst logisch und überlegt vorgehen.

Nehmen wir an, Sie finden irgendwo einen Schlüssel. Sie denken ´prima, doch was soll ich damit ?´
Im wirklichen Leben würden Sie ihn sich ansehen, würden ihn genau untersuchen und dann entscheiden, ob Sie ihn

liegenlassen oder ob Sie ihn für brauchbar oder sogar wertvoll halten und daher mitnehmen. Diese einzelnen Schritte müssen Sie im Adventure unbedingt nachvollziehen.

Geben Sie ein: UNTERSUCHE SCHLÜSSEL. Als Antwort erhalten Sie vielleicht: ES IST DER SCHLÜSSEL ZU MEINER WOHNUNG oder: ES IST EIN AUTOSCHLÜSSEL oder: ER IST AUS PUREM GOLD. Natürlich würden Sie keinen dieser Schlüssel liegen lassen, sondern alle mitnehmen, weshalb Sie auch: NIMM SCHLÜSSEL eingeben. Auf dem Bildschirm erscheint als Mitteilung zumindest ein O.K. (Dieses O.K. werden Sie oft sehen, es bedeutet, daß der Adventureinterpreter Ihre Eingabe verstanden und ausgeführt hat.), wenn nicht sogar weitere Reaktionen folgen.
Natürlich wird der Schlüssel nun nicht mehr in der Kopfzeile ICH SEHE erscheinen, denn da Sie ihn eingesteckt haben, befindet er sich jetzt in Ihrer Manteltasche oder ähnlichem, jedoch nicht mehr im Raum.

Sollten sich auf diese Art und Weise zahlreiche Gegenstände in Ihren Taschen angehäuft haben, verlieren Sie vermutlich die Übersicht, weshalb Ihnen auch die Möglichkeit,

INVENTUR

zu machen geboten wird. Immer wenn Sie wissen wollen, ob Sie nicht zuviel unnützes Zeug mit sich herumtragen, oder was Schuld daran war, daß Sie im Treibsand versunken sind, oder aber, welche Gegenstände geeignet wären, eine bestimmte Aufgabe zu erfüllen, so geben Sie einfach INV ein.

Als Antwort erscheint sofort: *ICH TRAGE FOLGENDES MIT MIR:* sowie eine Liste all dieser Objekte.

SPIELUNTERBRECHUNG:

Sie werden es wohl kaum schaffen, ein Adventure in einem Anlauf zu lösen, und natürlich wollen Sie am nächsten Tag nicht wieder von vorne beginnen.
Daher erlauben es fast alle dieser Spiele, den augenblicklichen Spielstand abzuspeichern.

Wenn Sie ein Spiel vorläufig beenden wollen, so geben Sie einfach *SAVE* ein. Bessere Programme fragen Sie dann nach einem Namen, unter dem die augenblickliche Situation abgespeichert werden soll, einfache erfüllen diese Aufgabe sofort.

Dieses Abspeichern empfiehlt sich auch zwischendurch: So könnte beispielsweise die Situation auftreten, daß Sie mit falschem Schuhwerk eine steile Felswand besteigen wollen. Hier ist die Wahrscheinlichkeit sehr groß, daß Sie abstürzen und sich Ihr Genick brechen; als Folge davon müßten Sie von vorne beginnen und alle Eingaben wiederholen.
Vor dieser uninteressanten Tätigkeit hätte Sie ein rechtzeitiges Abspeichern bewahrt !

FORTSETZUNG EINES UNTERBROCHENEN SPIELS:

Natürlich müssen Sie zunächst die entsprechende Cassette / Diskette einlegen, dann geben Sie einfach *LOAD* ein. Eventurell werden Sie gefragt, unter welchem Namen Sie den Spielstand abgespeichert hatten, andernfalls reicht das Drücken der <RETURN> - Taste allein, um das Spiel an der unterbrochenen Stelle fortzusetzen.

SONSTIGE HINWEISE:

Wenn das Titelbild Sie nicht entsprechend informiert, müssen Sie zunächst herausfinden, welchem Typ das geladene Adventure angehört.
Entweder haben Sie eine Aufgabe zu lösen, einen Ausgang zu suchen oder Schätze zu sammeln.
Untersuchen Sie alles, was Sie sehen, und verhalten Sie sich immer logisch (so brauchen Sie zum ÖFFNEn einer VERSCHLOSSENEn TÜR zunächst einen SCHLÜSSEL, eine BRECHSTANGE o. ä.), zumindest solange, bis Sie darüber informiert werden, daß Magie im Spiel ist. Dann dürfen Sie auch völlig sinnlose Eingaben machen und darauf hoffen, möglichst schnell auf die richtige Kombination zu kommen.
Achten Sie vor allem auf Mitteilungen der Art DAS GEHT IM MOMENT NICHT, denn damit soll Ihnen deutlich gemacht werden, daß Sie sich auf dem richtigen Weg befinden.
Leider sind aber zur Durchführung der Aktion noch nicht alle Bedingungen erfüllt (z. B. fehlt der SCHLÜSSEL).

Sollte ein Verb oder das Objekt nicht verstanden werden, so wird Ihnen darüber Mitteilung gemacht, und Sie sollten versuchen, die gleiche Aktion mit anderen Worten zu umschreiben.

Eine letzte Möglichkeit wird durch ICH VERSTEHE NICHT, WAS DU MEINST ausgedrückt. Die von Ihnen benutzten Worte sind zwar bekannt, geben in Ihrer Zusammenstellung jedoch keinen Sinn. Oder aber, die geschilderte Handlung wurde vom Programmierer des vorliegenden Adventures nicht vorgesehen.

TIPS ZUR LÖSUNG EINES ADVENTURES

Wenn Sie in verfahrenen Situationen überhaupt nicht mehr weiter wissen, können Sie es zunächst mit HELP versuchen.

Gelangen Sie auf diese Weise nicht in den Besitz weiterer Informationen, rufen Sie sich alle bisher entdeckten Gegenstände in die Erinnerung zurück und gehen alle Kombinationsmöglichkeiten mit den bislang bekannten Verben durch. Vielleicht stoßen Sie so auf eine erfolgversprechende Aktion, denn es kommt mit ziemlicher Sicherheit jedem Objekt eine Aufgabe zu; es nur zur Dekoration eines Raumes einzusetzen, dazu wäre der Programmieraufwand zu groß.

Für diese Aufgabe hilfreich ist in jedem Falle eine Liste aller vom Programm verstandenen Worte; möglicherweise wird diese sogar vom Adventure selbst auf den Befehl VOKabeln / VOCabulary hin erstellt.

Als ebenso nützlich erweist sich eine Karte, auf der Sie alle Räume mit Ihren Verbindungen eintragen, wobei Sie natürlich auch die Fundorte der Gegenstände nicht vergessen.

Sollten Sie unverhofft, oder auch gewollt, weil unvermeidlich, in ein Labyrinth geraten, so ist die Erstellung einer solchen Karte gar nicht so einfach, wie Sie es sich jetzt vielleicht vorstellen.
Denn statt wirklich vorhandener Räume hat man oft die dreifache Anzahl kartographiert.
Der einzige Trick, um dieses zu verhindern, besteht darin, in jedem Raum des Irrgartens einen Gegenstand aus dem Inventory abzulegen und jedem einzelnen Aufenthaltsort damit ein unverwechselbares Kennzeichen zu geben.

Und den wichtigsten Tip noch einmal:
Speichern Sie Ihr Spiel zwischendurch immer wieder ab !

Das verzauberte Schloss

Dieses Adventure möchte ich Ihnen besonders ans Herz legen, hier wird besonders tief in die Trickkiste gegriffen, um dem Abenteurer einige Überraschungen bieten zu können.
Geplant ist es eigentlich, daß der Spieler sich zunächst über die gestellte Aufgabe klar werden muß, ein Problem, welches Sie bedauerlicherweise bereits während des Eintippens lösen werden.

Zahllose der in diesem Buch vorgestellten Techniken wurden verwandt, einige Programmteile sollen hier als Beispiele genannt werden.
Flags wurden hier verwandt, um eine genaue Kontrollmöglichkeit der verschiedenen Stadien vor Verlassen des Schlosses zu haben.
Sind alle Voraussetzungen erfüllt, wird ein entsprechender Durchgang geöffnet, und die Adventurewelt vergrößert sich um 16 Räume.

```
1  REM Schlossadventure, Version 1.2
2  REM (c) 1984
3  REM by  Walkowiak
4  REM ********************************
11 CLS:LOCATE 9,4:PRINT"Das verzauberte
Schloss"
14 LOCATE 12,20:PRINT"ein Adventure von"
15 LOCATE 14,21:PRINT"J";CHR$(178);"rg W
alkowiak"
16 LOCATE 3,25:PRINT CHR$(164);" 1984  b
y DATA BECKER, Duesseldorf"
30 PLOT 261, 231:DRAW 241, 231:DRAW 241,
 221:DRAW 221, 221:DRAW 221, 231:DRAW 20
1, 231:PLOT 120, 280:DRAW 140, 260:DRAW
140, 120:DRAW 200, 120:DRAW 200, 260:DRA
W 220, 280:DRAW 120, 280:PLOT 200, 220:D
RAW 420, 220
31 PLOT 420, 120:DRAW 420, 260:DRAW 400,
 280:DRAW 500, 280:DRAW 480, 260:DRAW 48
0, 120:DRAW 420, 120:PLOT 419, 122:DRAW
201, 122:PLOT 292, 122:DRAW 292, 146:DRA
W 300, 154:DRAW 308, 154:DRAW 316, 146:D
RAW 316, 122
32 PLOT 400, 281:PLOT 400, 281:DRAW 400,
 291:DRAW 420, 291:DRAW 420, 281:DRAW 44
0, 281:DRAW 440, 291:DRAW 460, 291:DRAW
460, 281:DRAW 480, 281:DRAW 480, 291:DRA
W 500, 291:DRAW 500, 281:PLOT 220, 281:D
RAW 220, 291
33 DRAW 200, 291:DRAW 200, 281:DRAW 180,
 281:DRAW 180, 291:DRAW 160, 291:DRAW 16
0, 281:DRAW 140, 281:DRAW 140, 291:DRAW
120, 291:DRAW 120, 281:PLOT 172, 240:DRA
W 169, 237:DRAW 169, 228:DRAW 175, 228:D
RAW 175, 237
34 DRAW 172, 240:PLOT 175, 183:DRAW 175,
 192:DRAW 172, 195:DRAW 169, 192:DRAW 16
9, 183:PLOT 169, 180:DRAW 175, 180:PLOT
448, 180:DRAW 454, 180:DRAW 454, 189:DRA
W 451, 192:DRAW 448, 189:DRAW 448, 180:P
LOT 448, 228
```

```
35 DRAW 448, 237:DRAW 451, 240:DRAW 454,
 237:DRAW 454, 228:DRAW 448, 228:PLOT 42
1, 231:DRAW 401, 231:DRAW 401, 221:DRAW
381, 221:DRAW 381, 231:DRAW 361, 231:DRA
W 361, 221:DRAW 341, 221:DRAW 341, 231:D
RAW 321, 231
36 DRAW 321, 221:DRAW 301, 221:DRAW 301,
 231:DRAW 281, 231:DRAW 281, 221:DRAW 26
1, 221:DRAW 261, 231:DRAW 241, 231:DRAW
241, 221:DRAW 221, 221:DRAW 221, 231:DRA
W 201, 231:PLOT 398, 180:DRAW 398, 189:D
RAW 395, 192
37 DRAW 392, 189:PLOT 392, 189:DRAW 392,
 180:DRAW 398, 180:PLOT 350, 180:DRAW 35
6, 180:DRAW 356, 189:DRAW 353, 192:DRAW
350, 189:DRAW 350, 180:PLOT 308, 180:DRA
W 314, 180:DRAW 314, 189:DRAW 311, 192:D
RAW 308, 189
38 DRAW 308, 180:PLOT 275, 180:DRAW 275,
 189:DRAW 272, 192:DRAW 269, 189:DRAW 26
9, 180:DRAW 275, 180:PLOT 236, 180:DRAW
236, 189:DRAW 233, 192:DRAW 230, 189:DRA
W 230, 180:DRAW 236, 180:PLOT 299, 150:D
RAW 299, 129
39 PLOT 305, 129:DRAW 305, 153:PLOT 311,
 153:DRAW 311, 129:PLOT 317, 132:DRAW 29
3, 132
40 FOR i=1 TO 8000:NEXT
150  ar= 26:ao= 45:av= 9:af= 3
160  spieler= 9
170  wl= 4
190  DIM raum$(ar),durchgang(ar,6),ob$(a
o),rufname$(ao),ob(ao),verb$(av)
200  REM ************************** VERBE
N
210  DATA untersuche
211  DATA nimm
212  DATA lies
213  DATA oeffne
214  DATA zerstoere
215  DATA benutze
```

```
216  DATA fuelle
217  DATA leg
218  DATA wecke
300  REM ******************** GEGENSTAEND
E
301  DATA unzaehlige Buecherregale,regal
,1
302  DATA einen Tisch,tisch,1
303  DATA meinen Lehrmeister,Lehrer,2
304  DATA einen grossen Topf,Topf,3
305  DATA eine Flasche,Flasche,0
306  DATA den Schlossbrunnen,Brunnen,4
307  DATA ein Buch,Buch,0
308  DATA schlafende Waechter,Waechter,5
309  DATA den Seilzug der Zugbruecke,Sei
lzug,5
310  DATA Bewohner des Schlosses,Bewohne
r,8
311  DATA meine Gaeste,Gaeste,8
312  DATA einen Zettel,Zettel,0
313  DATA einen Kuechentisch,Kuechentisc
h,3
314  DATA ein Schlachtermesser,Messer,0
315  DATA eine Armbrust,Armbrust,5
316  DATA einen Schluessel,Schluessel,0
317  DATA ein schweres Eisentor,Tor,6
318  DATA eine Zugbruecke,Zugbruecke,7
319 DATA ,,0
320 DATA ,,0
321  DATA Baeume,Baeume,12
322  DATA Baeume,,13
323  DATA mehrere vollgepackte Regale,Re
gal,14
324 DATA ,,0
325  DATA Baeume,Baeume,15
326  DATA Baeume,,16
327  DATA Baeume,,17
328  DATA eine grosse Elster,Elster,17
329  DATA einen Hoehleneingang,Hoehle,18
330  DATA golden glaenzendes Metall,Meta
ll,19
```

```
331  DATA eine alte Ruestung,Ruestung,0
332  DATA nichts besonderes,,21
333  DATA einen Sumpf,Sumpf,0
334  DATA Schlamm,Schlamm,23
335  DATA Schlangeneier,Eier,26
336  DATA einen truegerischen Boden,Bode
n,23
337  DATA sprudelndes Wasser,Wasser,24
338  DATA einen Schlossgraben,Graben,10
339  DATA nichts besonderes,,20
340  DATA Spinnweben,Spinnweben,0
341  DATA eine Flasche mit blauem Wasser
,Flasche,0
342  DATA ,,0
343  DATA eine Schlangengrube,Grube,22
344  DATA Schlangen,Schlangen,26
345  DATA einen steinernen Altar,Altar,2
5
500  REM ************* RAUMBESCHREIBUNGE
N
501  DATA in der Schlossbibliothek,0,3,0
,2,0,0
502  DATA im Studierzimmer,0,0,1,0,0,0
503  DATA in der Kueche,1,8,0,4,0,0
504  DATA im Schlosshof,0,8,3,6,0,0
505  DATA in einem der Wachttuerme,0,0,0
,0,0,11
506  DATA im Schlosshof,11,0,4,0,0,0
507  DATA vor der Zugbruecke,0,0,6,0,0,0
508  DATA im grossen Festsaal,4,8,3,9,0,
0
509  DATA in einem Privatgemach,9,9,8,0,
0,0
510  DATA auf einer Zugbruecke,0,0,7,12,
0,0
511  DATA im Schlosshof,0,6,0,0,5,0
512  DATA auf einem Waldweg,12,16,10,13,
0,0
513  DATA im Zauberwald,21,17,12,15,0,0
514  DATA in einer duesteren Hoehle,0,18
,14,0,0,0
```

```
515  DATA im Wald,15,15,15,16,0,0
516  DATA im Zauberwald,12,21,15,17,0,0
517  DATA im Zauberwald,13,22,16,18,0,0
518  DATA am Fusse eines Gebirges,14,0,1
7,19,0,0
519  DATA in einer duesteren Hoehle,25,0
,18,19,0,0
520  DATA in einem Schloss,20,20,20,21,0
,0
521  DATA im Wald,16,21,20,21,0,0
522  DATA im Sumpf,17,0,0,23,0,0
523  DATA im Sumpf,0,0,22,24,0,0
524  DATA an einer Quelle,23,24,0,23,0,0
525  DATA in einer duesteren Hoehle,25,1
9,25,14,14,0
526  DATA in der Schlangengrube,0,0,0,0,
22,0
600 m$(0)="Ich verstehe nicht was Du mei
nst !"
601  m$(1)="Ich sehe nichts besonderes."
602  m$(2)="So stark bin ich nicht !"
603 m$(3)="Sie scheinen tief zu schlafen
."
604 m$(4)="Das halte ich fuer wenig sinn
voll."
605 m$(6)="Die Elster hat mir etwas gest
ohlen."
606 m$(7)="Da liegt eine alte Ruestung."
607 m$(8)="In der Hand haelt er einen Ze
ttel."
609 m$(9)="In einer Tasche ist ein Schlu
essel."
610 m$(10)="Eine Inschrift besagt:
             Nimm unser Opfer und sei uns
 gnaedig."
690  ok$="Okay !"
699  REM *********** hier evtl. 2. Titel
800  FOR i=1 TO av
810  READ verb$(i):verb$(i)=LEFT$(verb$(
i),wl):verb$(i)=UPPER$(verb$(i))
820  NEXT i
```

```
830  FOR objekt=1 TO ao
840  READ ob$(objekt),rufname$(objekt),o
b(objekt):rufname$(objekt)=LEFT$(rufname
$(objekt),wl):rufname$(objekt)=UPPER$(ru
fname$(objekt)):NEXT objekt
850  FOR raum=1 TO ar:READ raum$(raum)
860  FOR richtung=1 TO 6:READ durchgang(
raum,richtung)
870  NEXT richtung:NEXT raum
880  PRINT:INPUT"   Wuenschen Sie Ratsch
laege fuer Ihr                weiteres Vo
rgehen";eingabe$:eingabe$=UPPER$(eingabe
$)
890  IF LEFT$(eingabe$,1)="J"THEN GOSUB
900 ELSE GOTO 1000
895 GOTO 1000
900  MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,25
910  PRINT"       CPC - Ventures":PRINT T
AB(15)CHR$(164);" 1984  by J"CHR$(178);"
rg Walkowiak
920  PRINT STRING$(40,208):PRINT"Stellen
 Sie  sich einen Roboter vor, denSie  mi
t zahlreichen  Kommandos  steuernkoennen
."
930  PRINT"Ich bin dieser  Roboter,  und
 ich  werdemich fuer Sie den Gefahren de
r verwegen-sten Abenteuer aussetzen."
940  PRINT"Damit Sie  mich sinvoll  agie
ren  lassenkoennen,  werde ich  Ihnen di
e Situationin der ich  mich gerade befin
de, jeweils genau beschreiben."
950  PRINT"Anschliessend  sagen  Sie  mi
r  mit zweiWorten,  wie  beispielsweise
 UNTERSUCHETUER, NIMM MESSER, was ich tu
n soll.":PRINT
960  PRINT"Darueber hinaus verstehe ich
die Befehle      INVENTUR, SAVE, LOAD und
 ENDE."
970  PRINT STRING$(40,210):INK 3,12,24:I
NK 2,24,12:PEN 3: PRINT"            <Tas
```

```
te>";:PEN 2:PRINT" druecken";:PEN 1:PRIN
T
980  eingabe$=INKEY$:IF eingabe$="" THEN
 980 ELSE CLS:MODE 1:INK 1,2:INK 2,14:IN
K 3,26:RETURN
990  MODE 1:PAPER 0:INK 0,0:BORDER 0:INK
 1,2:INK 2,14:INK 3,26:PEN 1
1000  leerzeile$=STRING$(40,32)
1010  DATA Norden,Sueden,Westen,Osten,ob
en,unten
1020  FOR richtung=1 TO 6
1030  READ richtung$(richtung)
1040  NEXT richtung
1070  CLS
1080  PRINT:PRINT
1090  LOCATE 1,1
1100  FOR zeile=1 TO 10
1110  PRINT leerzeile$;
1120  NEXT zeile
1130  LOCATE 1,1:PEN 1
1140  PRINT"Ich bin ";
1150  PRINT raum$(spieler);:PRINT"."
1160  PRINT"Ich sehe ";:gedruckt=0
1170  FOR i=1 TO ao
1180  IF ob(i)<>spieler THEN 1210
1190  IF POS(#0)+LEN(ob$(i))+2<=39 THEN
PRINT ob$(i);", ";:GOTO 1205
1200  IF POS(#0)+LEN(ob$(i))+2>39 THEN P
RINT:GOTO 1190
1205  gedruckt=-1
1210  NEXT i
1215  IF NOT gedruckt THEN PRINT"nichts
besonderes  ";
1220  PRINT CHR$(8);CHR$(8);"."
1230  PRINT leerzeile$:PEN 2
1240  PRINT"Ich kann nach ";:gedruckt=0
1250  FOR richtung=1 TO 6
1260  IF durchgang(spieler,richtung)=0 T
HEN GOTO 1300 ELSE gedruckt=-1
1270  IF POS(#0)=15 THEN PRINT richtung$
(richtung);:GOTO 1300
```

```
1280  IF POS(#0)+LEN(richtung$(richtung)
)<38 THEN PRINT", ";richtung$(richtung);
:GOTO 1300
1290  PRINT",":PRINT richtung$(richtung)
;:GOTO 1300
1300  NEXT richtung
1310  IF gedruckt=0 THEN PRINT"nirgendwo
";
1320  PRINT".":PEN 3
1330  PRINT STRING$(40,154)
1340 IF spieler=25 THEN GOSUB 14000
1390  PEN 3:LOCATE 1,25:INPUT"Was soll i
ch tun";eingabe$:eingabe$=UPPER$(eingabe
$):PEN 2
1392 IF spieler=17 THEN GOSUB 13000
1394 IF spieler=23 THEN GOSUB 15100
1396 IF spieler=15 THEN GOSUB 15200
1400  IF LEN(eingabe$)>2 THEN 1500
1410  IF eingabe$="N" AND durchgang(spie
ler,1)<>0 THEN spieler=durchgang(spieler
,1):PRINT ok$:GOTO 1080
1420  IF eingabe$="S" AND durchgang(spie
ler,2)<>0 THEN spieler=durchgang(spieler
,2):PRINT ok$:GOTO 1080
1430  IF eingabe$="W" AND durchgang(spie
ler,3)<>0 THEN spieler=durchgang(spieler
,3):PRINT ok$:GOTO 1080
1440  IF eingabe$="O" AND durchgang(spie
ler,4)<>0 THEN spieler=durchgang(spieler
,4):PRINT ok$:GOTO 1080
1450  IF eingabe$="OB" AND durchgang(spi
eler,5)<>0 THEN spieler=durchgang(spiele
r,5):PRINT ok$:GOTO 1080
1460  IF eingabe$="U" AND durchgang(spie
ler,6)<>0 THEN spieler=durchgang(spieler
,6):PRINT ok$:GOTO 1080
1470  IF LEN(eingabe$)<3 THEN PRINT"Dahi
n fuehrt kein Weg !":GOTO 1080
1480  IF LEN(eingabe$)>8 THEN GOTO 2000
1499  REM ***************** START INVENTU
R
```

```
1500  IF LEFT$(eingabe$,3)<>"INV" THEN G
OTO 1600
1510  PRINT"Ich trage folgendes mit mir:
"
1520  FOR objekt=1 TO ao
1530  IF ob(objekt)=-1 THEN PRINT ob$(ob
jekt)
1540  NEXT objekt
1550  GOTO 1080
1560  REM ***************** ENDE INVENTUR
1600  IF LEFT$(eingabe$,3)<>"SAV" THEN G
OTO 1700
1610  PRINT"REC & PLAY druecken ...
             Unter welchem Namen speicher
n ..........";STRING$(10,8);:INPUT einga
be$
1620  IF LEN(eingabe$)>10 THEN PRINT"Bit
te etwas kuerzer !":GOTO 1610 ELSE einga
be$="!"+eingabe$+".spiel":OPENOUT eingab
e$
1630  PRINT#9,spieler
1640  FOR objekt=1 TO ao
1650  PRINT#9,ob(objekt)
1660  NEXT objekt
1670  FOR raum=1 TO ar:FOR richtung=1 TO
 6:PRINT#9,durchgang(raum,richtung):NEXT
 richtung:NEXT raum
1680  FOR flag=1 TO af:PRINT#9,fl)flag:N
EXT flag
1690  CLOSEOUT:PRINT ok$:GOTO 1080
1700  IF LEFT$(eingabe$,3)<>"LOA" THEN G
OTO 1900
1710  PRINT"Cassette rueckspulen & PLAY
druecken ...Welches Spiel laden ........
..";STRING$(10,8);:INPUT eingabe$
1720  IF LEN(eingabe$)>10 THEN PRINT"Das
 kann nicht sein !":GOTO 1710 ELSE einga
be$="!"+eingabe$+".SPIEL":OPENIN eingabe
$
1730  INPUT#9,spieler
1740  FOR objekt=1 TO ao:INPUT#9,ob(obje
```

```
kt):NEXT objekt
1750  FOR raum=1 TO ar:FOR richtung=1 TO
 6:INPUT#9,durchgang(raum,richtung):NEXT
 richtung:NEXT raum
1760  FOR flag=1 TO af:INPUT#9,fl(flag):
NEXT flag
1770  CLOSEIN:PRINT ok$:GOTO 1080
1900  IF LEFT$(eingabe$,3)<>"INS" THEN G
OTO 1950
1910  GOSUB 900:GOTO 1080
1950  IF LEFT$(eingabe$,3)<>"END" THEN G
OTO 2000
1960  PRINT"Der Autor wuenscht Ihnen meh
r Erfolg    beim naechsten Mal.":PRINT:P
RINT:PRINT:END
2000  laenge=LEN(eingabe$)
2010  FOR buchstabe=1 TO laenge
2020  pruef$=MID$(eingabe$,buchstabe,1)
2030  IF pruef$<>" "THEN NEXT buchstabe
2040  everb$=LEFT$(eingabe$,wl)
2050  rl=laenge-buchstabe
2060  IF rl<0 THEN 2090
2070  eobjekt$=RIGHT$(eingabe$,rl)
2080  eobjekt$=LEFT$(eobjekt$,wl)
2090  FOR verbnummer=1 TO av
2100  IF everb$=verb$(verbnummer) THEN 2
130
2110  NEXT verbnummer
2120  PRINT"Das Verb verstehe ich nicht.
":GOTO 1080
2130  FOR o=1 TO ao
2140  IF eobjekt$=rufname$(o) THEN 2200
2150  NEXT o
2160  PRINT"Diesen Gegenstand kenne ich
nicht.":GOTO 1080
2200  ON verbnummer GOTO 5000, 6000, 700
0, 8000, 9000, 10000, 11000, 12000, 1600
0
4500  CLS:REM SPIELER TOD
4510  PRINT"Auch das noch !":PRINT:PRINT
 m$(0)
```

```
4520  PRINT:PRINT"Ich bin tod !":PRINT
4530  INPUT"Soll ich es noch einmal vers
uchen ";eingabe$:eingabe$=UPPER$(eingabe
$)
4540  IF LEFT$(eingabe$,1)="J" THEN RUN
4550  GOTO 1960
4800  CLS:REM SPIELER SIEGT
4810  PRINT"Herzlichen Glueckwunsch !"
4820  PRINT:PRINT"Sie haben die Ihnen ge
stellte Aufgabe":PRINT:PRINT"geloest und
 duerfen sich nun an einem":PRINT:PRINT"
anderen Adventure versuchen."
4830  PRINT:PRINT:PRINT:PRINT:END
5000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 5900
5001 IF o=16 AND spieler=20 AND ob(31)<>
-1 AND ob(40)<>-1 THEN PRINT m$(7):ob(31
)=20:ob(40)=20:GOTO 1080
5002 IF o=10 AND spieler=8 THEN PRINT m$
(3):GOTO 1080
5003 IF o=11 AND spieler=7 THEN PRINT m$
(3):GOTO 1080
5004 IF o=4 AND (spieler=3 OR ob(4)=-1)
THEN PRINT m$(1):GOTO 1080
5005 IF o=5 AND (spieler=3 OR ob(o)=-1)
THEN PRINT m$(1):GOTO 1080
5006 IF o=1 AND spieler=1 THEN PRINT m$(
1):GOTO 1080
5008 IF o=3 AND spieler=2 AND ob(12)=0 T
HEN PRINT m$(8):ob(12)=2:GOTO 1080
5009 IF o=3 AND spieler=2 THEN PRINT m$(
1):GOTO 1080
5010 IF o=6 AND ob(5)=0 THEN PRINT"Auf d
em Wasser treibt eine Flasche.":ob(5)=sp
ieler:GOTO 1080
5011 IF o=7 AND (spieler=1 OR ob(o)=-1)
THEN PRINT"Es ist ein Buch ueber Zaubert
raenke.":GOTO 1080
5012 IF o=8 AND spieler=5 AND ob(16)=0 T
HEN PRINT m$(9):ob(16)=spieler:GOTO 1080
5013 IF o=9 AND NOT fl(1) THEN PRINT m$(
```

```
1):GOTO 1080
5014 IF o=9 AND fl(1) AND spieler=5 THEN
 PRINT"Er ist nicht funktionsfaehig.":GO
TO 1080
5015 IF o=12 AND ob(12)=-1 THEN PRINT"Da
rauf steht etwas geschrieben.":ob(16)=sp
ieler:GOTO 1080
5016 IF o=12 AND ob(12)=2 THEN PRINT"Daz
u muss ich ihn erst nehmen.":GOTO 1080
5017 IF o=13 AND ob(14)=0 THEN PRINT"Auf
 dem Tisch liegt ein Messer.":ob(14)=3:G
OTO 1080
5018 IF o=2 AND (ob(7)=1 OR ob(7)=0) THE
N PRINT"Darauf liegt ein altes Buch.":ob
(7)=1:GOTO 1080
5019 IF o=14 AND (ob(14)=spieler OR ob(1
4)=-1) THEN PRINT"Die Klinge ist sehr sc
harf.":GOTO 1080
5020 IF o=15 AND (ob(o)=spieler OR ob(o)
=-1) THEN PRINT m$(1):GOTO 1080
5021 IF o=16 AND (ob(16)=-1 OR ob(16)=sp
ieler) THEN PRINT m$(1):GOTO 1080
5022 IF o=17 AND spieler=6 AND fl(3)<>-1
 THEN PRINT"Es ist verschlossen.":GOTO 1
080
5023 IF o=17 AND spieler=6 AND fl(3)=-1
THEN PRINT"Es steht weit offen.":GOTO 10
80
5024 IF o=18 AND spieler=7 AND fl(1) THE
N PRINT"Die Zugbruecke faellt nach unten
.":durchgang(7,4)=10:GOTO 1080
5025 IF o=18 AND spieler=7 AND NOT fl(1)
 THEN PRINT"Die Zugbruecke ist hochgezog
en.":GOTO 1080
5026 IF o=21 THEN PRINT m$(1):GOTO 1080
5027 IF o=23 AND spieler=14 THEN m0$="Es
 war eine Zauberkueche.             De
r Zauberer kam und machte einen Stein au
s mir.":GOTO 4500
5028 IF o=28 AND spieler=17 THEN PRINT"E
s ist wirklich ein Prachtexemplar.":GOSU
```

```
B 13000:GOTO 1080
5029 IF o=29 AND spieler=18 THEN PRINT"E
in merkwuerdiger Geruch dringt heraus.";
:GOTO 1080
5030 IF o=30 AND (spieler=19 OR ob(30)=-
1) THEN PRINT"Es ist wertloses Metall.":
GOTO 1080
5031 IF o=31 AND (spieler=20 OR ob(31)=-
1) THEN PRINT m$(1):GOTO 1080
5032 IF o=35 AND (spieler=26 OR ob(35)=-
1) THEN PRINT m$(1):GOTO 1080
5033 IF o=36 AND spieler=23 THEN PRINT"E
r sieht nicht sehr tragfest aus.":GOTO 1
080
5034 IF o=38 AND spieler=10 THEN PRINT m
$(1):GOTO 1080
5035 IF o=37 AND spieler=24 THEN PRINT"E
s ist das beruehmte blaue Wasser.":GOTO
1080
5036 IF o=40 AND (spieler=20 OR ob(40)=-
1) THEN PRINT m$(1):GOTO 1080
5037 IF o=43 THEN spieler=26:PRINT"Okay
!":GOTO 1080
5038 IF o=45 THEN PRINT m$(10):GOTO 1080
5899 PRINT m$(1):GOTO 1080
5900 IF o=21 THEN PRINT m$(1):GOTO 1080
5901 IF o=16 AND spieler=20 AND ob(31)<>
-1 AND ob(40)<>-1 THEN PRINT m$(7):ob(31
)=20:ob(40)=20:GOTO 1080
5902 IF o=33 AND spieler=23 THEN GOSUB 1
3100:PRINT"Im Osten scheint eine Quelle
zu sein.":GOTO 1080
5903 IF o=16 AND (spieler=23 OR ob(34)=-
1) THEN PRINT"Es ist Heilschlamm.":GOSUB
 13100:GOTO 1080
5904 IF o=2 AND spieler=3 AND ob(14)=0 T
HEN PRINT"Auf dem Tisch liegt ein Messer
.":ob(14)=3:GOTO 1080
5905 IF o=2 AND spieler=3 AND ob(14)<>0
THEN PRINT m$(1):GOTO 1080
5990 PRINT"So etwas sehe ich hier nicht
```

```
!":GOTO 1080
6000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 6900
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6002 IF o=6 THEN PRINT m$(2):GOTO 1080
6003 IF o=17 THEN PRINT m$(2):GOTO 1080
6004 IF o=2 THEN PRINT m$(4):GOTO 1080
6005 IF o=3 THEN PRINT m$(4):GOTO 1080
6006 IF o=8 THEN PRINT m$(4):GOTO 1080
6007 IF o=10 THEN PRINT m$(4):GOTO 1080
6008 IF o=11 THEN PRINT m$(4):GOTO 1080
6009 za=0:FOR i=1 TO ao:IF ob(i)=-1 THEN
 za=za+1:IF za=4 THEN GOTO 6099
6010 NEXT i
6011 IF o=5 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6013 IF o=12 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6014 IF o=15 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6015 IF o=16 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6016 IF o=14 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6017 IF o=9 THEN PRINT m$(2):GOTO 1080
6018 IF o=23 THEN PRINT m$(2):GOTO 1080
6019 IF o=28 THEN PRINT"Womit soll ich s
ie fangen ?":GOSUB 13000:GOTO 1080
6020 IF o=18 THEN PRINT m$(0):GOTO 1080
6021 IF o=30 THEN ob(o)=-1:PRINT ok$:GOT
O 1080
6022 IF o=31 THEN ob(o)=-1:PRINT"Ich hab
e die Ruestung angelegt.":GOTO 1080
6023 IF o=35 AND ob(31)=-1 THEN PRINT ok
$:ob(35)=-1:GOTO 1080
6024 IF o=35 AND ob(31)<>-1 THEN m$(0)="
Eine Schlange hat mich gebissen.":GOTO 4
500
6025 IF o=36 THEN PRINT m$(4):GOTO 1080
```

```
6026 IF o=37 AND ob(5)=-1 THEN PRINT ok$
:ob(5)=0:ob(41)=-1:GOTO 1080
6027 IF o=37 AND ob(5)<>-1 THEN PRINT"Zu
vor brauche ich eine leere Flasche.":GOT
O 1080
6028 IF o=38 THEN PRINT m$(0):GOTO 1080
6029 IF o=40 THEN ob(40)=-1:PRINT ok$:GO
TO 1080
6032 IF o=13 THEN PRINT m$(4):GOTO 1080
6033 IF o=4 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6900 IF o=16 AND spieler=23 AND ob(4)<>-
1 THEN PRINT"Zuvor brauche ich ein Gefae
ss !":GOTO 1080
6901 IF o=16 AND spieler=23 AND ob(4)=-1
 THEN PRINT ok$:ob(34)=-1:GOTO 1080
6902 IF o=2 AND spieler=3 THEN PRINT m$(
4):GOTO 1080
6998 PRINT"So etwas sehe ich hier nicht.
":GOTO 1080
6999 PRINT"Tut mir leid - aber ich trage
 schon":PRINT"mehr als genug !":GOTO 108
0
7000 IF o=13 AND ob(12)<>-1 THEN PRINT"I
ch habe keinen Zettel.":GOTO 1080
7001 IF o=12 AND ob(o)=-1 THEN co=INT(RN
D(1)*100):PRINT"Ich sehe zittrig geschri
eben die Zahl:":PRINT co:GOTO 1080
7002 IF n=7 AND ob(7)=-1 THEN GOTO 7650
7003 IF n=7 AND ob(7)<>-1 THEN PRINT"Daz
u muss ich ihn erst haben.":GOTO 1080
7649 PRINT m$(0):GOTO 1080
7650 INPUT"Welche Seite ";se
7651 IF se<>co THEN PRINT m$(1):GOTO 108
0
7652 CLS:PRINT"RATSCHLAEGE FUER BESONDER
E NOTLAGEN:":PRINT STRING$(40,154):PRINT
7654 PRINT"Sollte es Dir angeschehen, da
ss die":PRINT"Deinen von einem Zauberer
boeser Ge-"
7655 PRINT"sinnung in den langen Schlaf
```

```
versetzt":PRINT"werden, so hoere meinen
Rat:":PRINT
7657 PRINT:PRINT"Verschaffe Dir die vier
 Zutaten zu":PRINT"Humbug's Heilwasser u
nd bringe Sie"
7659 PRINT"dem Zauberer jenseits des Zau
berwaldes.";
7660 PRINT:PRINT"Sollte er Dir wohl geso
nnen sein,":PRINT"so wird er Dir helfen.
"
7661 PRINT:PRINT"Ist er es nicht, so kan
nst Du nur":PRINT"noch auf einen Prinzen
 hoffen, der"
7663 PRINT"auf traditionelle Weise Deine
 Tochter,":PRINT"und mit ihr alle Bewohn
er des Schlosses";:PRINT"wachkuesst."
7666 eingabe$=INKEY$:IF eingabe$="" THEN
 7666 ELSE CLS:GOTO 1080
7900 PRINT m$(0):GOTO 1080
8000 IF o=17 AND ob(16)<>-1 THEN PRINT"W
omit ?":GOTO 1080
8001 IF o=17 AND ob(16)=-1 THEN PRINT ok
$:durchgang(6,4)=7:fl(3)=-1:GOTO 1080
8002 IF o=35 AND ob(35)=-1 THEN PRINT"Da
rin war eine zaehe Fluessigkeit.":GOTO 1
080
8099 PRINT m$(0):GOTO 1080
9000 IF o=9 AND ob(14)=-1 AND spieler=5
THEN PRINT"Die Halteseile sind zerschnit
ten.":fl(1)=-1:GOTO 1080
9001 IF o=9 AND ob(14)<>-1 AND spieler=5
 THEN PRINT"Womit ?":GOTO 1080
9999 PRINT m$(0):GOTO 1080
10000 IF ob(o)<>spieler AND ob(o)<>-1 TH
EN PRINT"So etwas sehe ich hier nicht.":
GOTO 1080
10001 IF o=9 THEN PRINT"Dazu sind mindes
tens zwei Mann":PRINT"erforderlich.":GOT
O 1080
10002 IF o=14 AND spieler=5 THEN PRINT"D
ie Halteseile sind zerschnitten.":fl(1)=
```

```
-1:GOTO 1080
10003 IF o=31 THEN PRINT"Okay - ich bin
in der Ruestung.":ob(31)=-1:GOTO 1080
10990 PRINT m$(0):GOTO 1080
11000 IF o=5 AND spieler=24 AND ob(5)=-1
 THEN PRINT ok$:ob(o)=0:ob(41)=-1:GOTO 1
080
11001 IF o=5 AND spieler<>24 AND ob(5)=-
1 THEN PRINT"Womit ?":GOTO 1080
11990 PRINT m$(0):GOTO 1080
12000 IF ob(o)<>-1 THEN GOTO 12900
12001 IF o=4 THEN PRINT ok$:ob(o)=spiele
r:GOTO 1080
12002 IF o=5 THEN PRINT ok$:ob(o)=spiele
r:GOTO 1080
12003 IF o=7 THEN PRINT ok$:ob(o)=spiele
r:GOTO 1080
12004 IF o=12 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12005 IF o=15 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12006 IF o=16 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12007 IF o=14 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12008 IF o=30 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12009 IF o=41 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12010 IF o=45 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12011 IF o=40 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12012 IF o=36 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12013 IF o=35 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12014 IF o=31 THEN PRINT ok$:ob(o)=spiel
er:GOTO 1080
12900 IF o=5 AND ob(12)=-1 THEN PRINT ok
$:ob(41)=spieler:GOTO 1080
```

```
12901 IF o=16 AND ob(34)=-1 THEN PRINT o
k$:ob(34)=spieler:GOTO 1080
12990 PRINT m$(0):GOTO 1080
13000 IF ob(30)=-1 THEN ob(30)=20:PRINT
m$(6):RETURN
13010 IF ob(35)=-1 THEN ob(35)=20:PRINT
m$(6):RETURN
13020 IF ob(5)=-1 THEN ob(5)=20:PRINT m$
(6):RETURN
13090 RETURN
14000 IF ob(34)<>25 THEN RETURN
14010 IF ob(35)<>25 THEN RETURN
14020 IF ob(41)<>25 THEN RETURN
14030 IF ob(40)<>25 THEN RETURN
14100 CLS:PRINT"Eine Stimme ertoent:":PR
INT:PRINT
14110 PRINT"Der grosse Humbug freut sich
 ueber ":PRINT
14120 PRINT"Deine Gaben. Er wird seine Z
auber-":PRINT
14130 PRINT"kraft fuer Dich einsetzen.":
PRINT:PRINT
14140 PRINT"Gehe nun nach Hause, denn do
rt wird":PRINT
14150 PRINT"bereits ein Fest vorbereitet
, um":PRINT:PRINT"das Erwachen aus dem l
angen Schlaf":PRINT:PRINT"zu feiern."
14160 GOTO 14160
15100 IF ob(31)<>-1 THEN RETURN
15110 m$(0)="Ich bin zu schwer und versi
nke.":GOTO 4500
15200 IF RND(1)>0.3 THEN RETURN
15210 m$(0)="Ich bin in eine Fallgrube g
estuerzt.":GOTO 4500
16000 IF o=10 AND spieler=8 THEN PRINT"E
s gelingt mir nicht.":GOTO 1080
16010 IF o=11 AND spieler=8 THEN PRINT"E
s gelingt mir nicht.":GOTO 1080
16020 PRINT m$(0):GOTO 1080
```

GOLDRAUSCH

Viele Worte werden zu diesem Adventure wohl kaum noch nötig sein. Allerdings handelt es sich hier um die vollständige Version, das Abenteuer wird erst richtig beginnen, wenn es Ihnen gelungen ist, das bislang bekannte Territorium zu verlassen.

Sie werden in die dunkle Welt einer alten Goldmine eindringen und es sich zweimal überlegen, ob Sie einen spontanen Einfall probieren, und damit wertvolle Zeit, die hier an der Brenndauer einer Lampe gemessen wird, verstreichen lassen wollen, oder ob Sie lieber auf Nummer Sicher gehen und nur durch irgendwelche Überlegungen gerechtfertigte Handlungen durchführen wollen.

```
1 REM ***********************************
2 REM * Goldrausch, Ver 1               *
3 REM * (c) 1984 by Walkowiak           *
4 REM ***********************************

5 DEFINT a-z:ON ERROR GOTO 1780
10 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN 1
:INK 1,25:PEN 2:INK 2,24
20 PLOT 420, 170:DRAW 420, 360:DRAW 640,
 360:DRAW 640, 340:DRAW 450, 340:DRAW 45
0, 170:DRAW 460, 180:DRAW 460, 340:PLOT
460, 200:DRAW 530, 270:PLOT 530, 340:DRA
W 530, 260:DRAW 540, 260:DRAW 540, 340:P
LOT 540, 260
21 DRAW 550, 270:DRAW 550, 340:PLOT 550,
 290:DRAW 600, 340:PLOT 640, 340:DRAW 30
0, 0:PLOT 440, 0:DRAW 640, 200:PLOT 640,
 200:PLOT 500, 200:DRAW 640, 200:PLOT 60
0, 160:DRAW 460, 160:PLOT 420, 120:DRAW
560, 120
22 PLOT 520, 80:DRAW 380, 80:PLOT 340, 4
0:DRAW 480, 40:PLOT 440, 0:DRAW 300, 0:P
LOT 540, 240:DRAW 640, 240:PLOT 580, 280
:DRAW 640, 280:PLOT 620, 320:DRAW 640, 3
20:PLOT 450, 170:DRAW 420, 170:PLOT 420,
 360
23 DRAW 440, 380:DRAW 640, 380:PLOT 430,
 370:DRAW 430, 380:DRAW 440, 390:DRAW 44
0, 400:PLOT 280, 150:DRAW 150, 150:DRAW
150, 200:DRAW 280, 200:DRAW 280, 150:DRA
W 310, 180:PLOT 310, 180:PLOT 280, 200:D
RAW 310, 230
24 DRAW 310, 180:PLOT 310, 230:DRAW 180,
 230:DRAW 150, 200:PLOT 150, 200:DRAW 16
0, 210:DRAW 270, 210:DRAW 280, 200:PLOT
270, 210:DRAW 290, 230:PLOT 270, 190:DRA
W 160, 190:DRAW 160, 160:DRAW 270, 160:D
RAW 270, 190
25 PRINT CHR$(150);STRING$(18,154);CHR$(
156)
26 PRINT CHR$(149);STRING$(18,32);CHR$(1
```

```
49)
27 PRINT CHR$(149);"    GOLDRAUSCH    ";
CHR$(149)
28 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
29 PRINT CHR$(149);"ein Adventure  von";
CHR$(149)
30 PRINT CHR$(149);"  J";CHR$(178);"rg W
alkowiak  ";CHR$(149)
31 PRINT CHR$(149);STRING$(18,32);CHR$(1
49)
32 PRINT CHR$(147);STRING$(18,154);CHR$(
153)
33 LOCATE 3,25:PRINT CHR$(164);" 1984  b
y DATA BECKER, Duesseldorf";
40 FOR i=1 TO 8000:NEXT
110 AR=31
120 ao=45
130 av=11
140 af=7
150 spieler=1
160 wortlaenge=4
170 wmax=60
171 wertung=0
180 zug=0
185 imax=4
186 lm=0:licht=-1:lw=1:ll=20
190 DIM raum$(ar),durchgang(ar,6),ob$(ao
),rufname$(ao),ob(ao),verb$(av)
200 REM *************************** VERBEN
201 DATA untersuche
202 DATA nimm
203 DATA "leg "
204 DATA oeffne
205 DATA benutze
206 DATA zerstoere
207 DATA zuende
208 DATA fuelle
209 DATA betrete
210 DATA loesche
211 DATA befestige
```

```
300 REM ******************* GEGENSTAENDE
301 DATA"viele grosse Baeume","Baeume",1
302 DATA"viele grosse Baeume","Baeume",2
303 DATA"einige Felsbrocken","Felsen",2
304 DATA"eine verfallene Holzhuette","Hu
ette",3
305 DATA"eine verschmutzte Korbflasche",
"Flasche",0
306 DATA"Honig","Honig",0
307 DATA"eine Holzkiste","Kiste",3
308 DATA"ein klappriges Regal","Regal",0
309 DATA"etwas Sprengstoff","Sprengstoff
",0
310 DATA"ein duesteres Erdloch","Erdloch
",0
311 DATA"eine rostige Eisentruhe","Truhe
",0
312 DATA"*Silbermuenzen*","Silber",0
313 DATA"eine duestere Felsenhoehle","Ho
ehle",5
314 DATA"einen grimmig dreinblickenden B
aeren","Baer",0
315 DATA"zahllose niedrige Buesche","Bue
sche",4
316 DATA"mehrere Eisenstangen";"Eisensta
nge",6
317 DATA"*Nuggets*","Nugget",5
318 DATA"eine Eisenstange","Eisen",0
319 DATA"eine Eisenkette","Kette",0
320 DATA verrottete Schienenstraenge,Sch
ienen,7
321 DATA ein Seil,Seil,0
322 DATA eine schwere Spitzhacke,Hacke,8
323 DATA eine alte Laterne,Laterne,8
324 DATA das Gangende,Gangende,9
325 DATA einen Eisenhaken,Haken,0
326 DATA einen Schacht,Schacht,9
327 DATA eine Bretterwand,Bretterwand,10
328 DATA Abbauschutt,Schutt,11
329 DATA alte Saecke,Saecke,0
330 DATA feuchte Steinwaende,Waende,13
```

```
331 DATA eine uebelriechende Fluessigkei
t,Fluessigkeit,17
332 DATA eine Leiche,Leiche,18
333 DATA *Silberklumpen*,Klumpen,0
334 DATA *Goldmuenzen*,Muenzen,0
335 DATA Spinnweben,Spinnweben,20
336 DATA eine goldene Wand,Wand,22
337 DATA ein Schild,Schild,22
338 DATA *Gold*,Gold,31
339 DATA ein Seeufer,Ufer,30
340 DATA den See,"See ",30
341 DATA goldene Steine,Steine,0
342 DATA Zuendschnur,Zuendschnur,0
343 DATA ein Luntero,Luntero,-1
344 DATA -,Paradies,0
345 DATA eine alte Lore,Lore,7
500 REM ************* RAUMBESCHREIBUNGEN
501 DATA"im Wald",1,1,1,2,0,0
502 DATA"im Wald",2,1,1,3,0,0
503 DATA"im Wald vor einem Felshang",0,4
,2,0,0,0
504 DATA"auf einer Waldlichtung",3,0,5,0
,0,0
505 DATA"auf einer Waldlichtung",0,0,6,4
,0,0
506 DATA"vor einem Bergwerksstollen.",7,
0,1,5,0,0
507 DATA im Eingangsstollen,8,6,0,0,0,0
508 DATA in einer Nische des Ganges,9,7,
8,10,0,0
509 DATA am Ende des Ganges,0,8,0,0,0,0
510 DATA in einem Seitengang,11,0,8,0,0,
0
511 DATA in einer alten Abbaustelle,0,10
,0,0,0,0
512 DATA in der Hoehle,0,5,0,13,0,0
513 DATA in der Hoehle,0,0,12,14,0,0
514 DATA in der Hoehle,0,16,13,15,0,0
515 DATA in der Hoehle,0,0,14,0,0,0
516 DATA auf einem Gang,14,0,0,17,0,0
517 DATA in einer Nebenhoehle,0,0,16,0,0
```

```
,0
518 DATA auf dem Boden des Schachtes,0,0
,19,0,0,0
519 DATA auf einem kurvenreichen Gang,24
,0,20,18,0,0
520 DATA in einem breiten Gang,0,0,21,19
,0,0
521 DATA in einer alten Abbaustrecke,22,
11,11,20,11,0
522 DATA in einem Felsendom,0,21,0,0,27,
0
523 DATA in einem unterirdischen Paradie
s,0,0,0,0,0,0
524 DATA auf einem kurvenreichen Gang,26
,19,24,24,24,24
525 DATA auf einem kurvenreichen Gang,27
,24,24,29,26,24
526 DATA auf einem kurvenreichen Gang,27
,24,26,27,27,24
527 DATA auf einem kurvenreichen Gang,25
,24,26,27,26,24
528 DATA auf einem kurvenreichen Gang,24
,26,26,27,0,0
529 DATA auf einem kurvenreichen Gang,0,
27,30,27,0,0
530 DATA vor einem unterirdischen See,28
,0,0,0,0,0
531 DATA in einer kleinen Hoehle,30,0,0,
0,0,0
600 REM ******************* MITTEILUNGEN
601 m$(1)="Ich sehe nichts besonderes."
602 m$(2)="So stark bin ich nicht !"
603 m$(3)="Wie stellst Du Dir das vor ?"
604 m$(4)="Der Baer greift sich den Honi
g, und"
605 m$(5)="verschwindet in der Tiefe der
 Hoehle."
690 ok$="O.k."
699 REM *********** 2. TITEL: EINLEITUNG
700 CLS:PEN 1:PRINT"           Herzlich Wi
llkommen zu":PRINT:PEN 2:PRINT TAB(15)CH
```

```
R$(34);"GOLDRAUSCH";CHR$(34):PRINT:PEN 1
705 PRINT STRING$(40,208)
710 PRINT"Auf  der Suche  nach  dem Glue
ck in  derneuen  Welt  begegneten Sie  v
or einigenTagen einem  alten  todkranken
 Mann, demSie in  seinen letzten  Stunde
n Beistandleisteten."
720 PRINT"Aus Dankbarkeit  berichtete er
 Ihnen vonseiner Goldmine und den dort v
ersteckteneResten seines Vermoegens."
730 PRINT"Zahllosen Gefahren  widerstand
en Sie aufdem  Weg  dorthin;  bald  werd
en Sie IhrZiel erreicht  haben  und  es
 wird sichzeigen,  ob  der  Alte  die Wa
hrheit ge-sprochen,  oder  im  Fiebertra
um geredethatte."
740 PRINT STRING$(40,210)
810 FOR i=1 TO av
815 READ verb$(i):verb$(i)=LEFT$(verb$(i
),wortlaenge):verb$(i)=UPPER$(verb$(i))
820 NEXT i
830 FOR objekt=1 TO ao
835 READ ob$(objekt),rufname$(objekt),ob
(objekt):rufname$(objekt)=LEFT$(rufname$
(objekt),wortlaenge):rufname$(objekt)=UP
PER$(rufname$(objekt))
840 NEXT objekt
845 FOR RAUM=1 TO AR
850 READ RAUM$(RAUM)
855 FOR RICHTUNG=1 TO 6
860 READ durchgang(raum,richtung)
865 NEXT richtung
870 NEXT raum
880 PRINT:INPUT"   Wuenschen Sie Ratschl
aege fuer Ihr                weiteres Vorg
ehen";eingabe$:eingabe$=UPPER$(eingabe$)
890 IF LEFT$(eingabe$,1)="J" THEN GOSUB
900
895 GOTO 1000
899 REM ******** 3. TITEL: INSTRUKTIONEN
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
```

```
1:INK 1,25
910 PRINT"       CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J"
;CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Stellen
Sie  sich einen Roboter vor, denSie  mit
 zahlreichen  Kommandos  steuernkoennen.
"
930 PRINT"Ich bin dieser  Roboter,  und
ich  werdemich fuer Sie den Gefahren der
 verwegen-sten Abenteuer aussetzen."
940 PRINT"Damit Sie  mich sinnvoll  agie
ren lassenkoennen,  werde ich  Ihnen die
 Situationin der ich  mich gerade befind
e, jeweilsgenau beschreiben."
950 PRINT"Anschliessend  sagen  Sie  mir
  mit zweiWorten,  wie  beispielsweise
UNTERSUCHETUER, NIMM MESSER, was ich tun
 soll.":PRINT
960 PRINT"Darueber hinaus verstehe ich d
ie Befehle    INVENTUR, SCORE, VOKABELN,
 HELP                SAVE & LOAD sowie ENDE
."
970 PRINT STRING$(40,210)
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT
"           <TASTE>";:PEN 2:PRINT" drueck
en";:PEN 1:PRINT" ..."
990 eingabe$=INKEY$:IF eingabe$="" THEN
990 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK
 3,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1010 leerzeile$=STRING$(40," ")
1020 DATA Norden, Sueden, Westen, Osten,
 Oben, Unten
1030 FOR richtung=1 TO 6
1040 READ richtung$(richtung)
1050 NEXT richtung
1070 CLS
1080 PRINT:PRINT:zug=zug+1:lw=lw+1
```

```
1084 IF lw=lm THEN GOSUB 3000
1085 IF wertung=wmax THEN GOTO 4800
1090 LOCATE 1,1
1091 REM ******************** KEIN LICHT
1092 IF licht=-1 THEN 1100
1093 PRINT"Ich weiss nicht genau, wo ich
 bin.":PRINT"Es ist zu dunkel, um etwas
zu sehen."
1095 PRINT:PRINT"Auch die Ausgaenge sehe
 ich nicht":PRINT"mehr !":GOTO 1330
1100 FOR zeile=1 TO 10
1110 PRINT leerzeile$;
1120 NEXT zeile
1130 LOCATE 1,1:PEN 1
1140 PRINT"Ich bin ";
1150 PRINT raum$(spieler);:PRINT"."
1160 PRINT"Ich sehe ";
1170 FOR i=1 TO ao
1180 IF ob(i)<>spieler THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+2<=39 THEN P
RINT ob$(i);", ";:GOTO 1210
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT:GOTO 1190
1210 NEXT i
1220 PRINT CHR$(8);CHR$(8);"."
1230 PRINT leerzeile$:PEN 2
1240 PRINT"Ich kann nach ";:gedruckt=0
1250 FOR RICHTUNG=1 TO 6
1260 IF durchgang(spieler,richtung)=0 TH
EN GOTO 1300 ELSE gedruckt=-1
1270 IF POS(#0)=15 THEN PRINT richtung$(
richtung);:GOTO 1300
1280 IF POS(#0)+LEN(richtung$(richtung))
<38 THEN PRINT", ";richtung$(richtung);:
GOTO 1300
1290 PRINT",":PRINT richtung$(richtung);
:GOTO 1300
1300 NEXT richtung
1310 IF gedruckt=0 THEN PRINT"nirgendwo"
;
1320 PRINT".":PEN 3
```

```
1330 PRINT"------------------------------
----------"
1340 IF wertung=wmax THEN GOTO 4800
1350 IF spieler=15 THEN m$(0)="Leider wa
r dort der Baer.":FOR i=1 TO 1000:NEXT i
:GOTO 4500
1360 IF lw<=0 THEN licht=0
1370 IF ex=zug AND spieler=20 THEN m$(0)
="rrrumms ! - Auch mich hats zerrissen."
:GOTO 4500
1380 IF spieler>18 AND ob(23)<>-1 THEN l
icht=0
1390 PEN 3:LOCATE 1,25:INPUT"Was soll ic
h tun";eingabe$:eingabe$=UPPER$(eingabe$
):PEN 2
1395 rl=lm-lw:IF (rl>0 AND rl<15) THEN P
RINT"in";lm-lw;"Zuegen stehe ich im Dunk
len !"
1400 IF LEN(eingabe$)>2 THEN 1500
1410 IF eingabe$="N" AND durchgang(spiel
er,1)<>0 THEN spieler=durchgang(spieler,
1):PRINT"O.k.":GOTO 1080
1420 IF eingabe$="S" AND durchgang(spiel
er,2)<>0 THEN spieler=durchgang(spieler,
2):PRINT"O.k.":GOTO 1080
1430 IF eingabe$="W" AND durchgang(spiel
er,3)<>0 THEN spieler=durchgang(spieler,
3):PRINT"O.k.":GOTO 1080
1440 IF eingabe$="O" AND durchgang(spiel
er,4)<>0 THEN spieler=durchgang(spieler,
4):PRINT"O.k.":GOTO 1080
1450 IF eingabe$="OB" AND durchgang(spie
ler,5)<>0 THEN spieler=durchgang(spieler
,5):PRINT"O.k.":GOTO 1080
1460 IF eingabe$="U" AND durchgang(spiel
er,6)<>0 THEN spieler=durchgang(spieler,
6):PRINT"O.k.":GOTO 1080
1470 IF LEN(eingabe$)<3 THEN PRINT"Dahin
 fuehrt kein Weg !":GOTO 1080
1480 IF LEN(eingabe$)>8 THEN GOTO 2000
1499 REM **************** START INVENTUR
```

```
1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1560
1510 PRINT"Ich trage folgendes mit mir:"
1520 FOR objekt=1 TO ao
1530 IF ob(objekt)=-1 THEN PRINT ob$(obj
ekt)
1540 NEXT objekt
1550 GOTO 1080
1560 IF LEFT$(eingabe$,3)<>"SCO" THEN GO
TO 1600
1561 PRINT"Von";wmax;"Punkten hast Du in
";zug;"Zuegen":PRINT wertung;"Punkte err
eicht !":PRINT"Das entspricht einem Schn
itt von";wertung/zug;"Punkten.":GOTO 108
0
1599 REM ********************** SAVE GAME
1600 IF LEFT$(eingabe$,3)<>"SAV" THEN GO
TO 1700
1610 PRINT"REC & PLAY druecken ...":PRIN
T"Unter welchem Namen speichern ........
..";STRING$(10,8);:INPUT eingabe$
1620 IF LEN(eingabe$)>10 THEN PRINT"Bitt
e etwas kuerzer !":GOTO 1610 ELSE eingab
e$="!"+eingabe$+".SPIEL":OPENOUT eingabe
$
1625 PRINT#9,spieler
1630 FOR objekt=1 TO ao
1631 PRINT#9,ob(objekt)
1632 NEXT objekt
1635 FOR raum=1 TO ar
1636 FOR richtung=1 TO 6
1637 PRINT#9,durchgang(raum,richtung)
1638 NEXT richtung
1639 NEXT raum
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1660 PRINT ok$
1670 GOTO 1080
1699 REM *********************** LOAD GAME
```

```
1700 IF LEFT$(eingabe$,3)<>"LOA" THEN GO
TO 1800
1710 PRINT"Cassette ruecksspulen & PLAY d
ruecken ...":PRINT"Welches Spiel laden .
.........";STRING$(10,8);:INPUT eingabe$
1720 IF LEN(eingabe$)>10 THEN PRINT"Das
kann nicht sein !":GOTO 1710 ELSE eingab
e$="!"+eingabe$+".SPIEL":OPENIN eingabe$
1725 INPUT#9,spieler
1730 FOR objekt=1 TO ao
1731 INPUT#9,ob(objekt)
1732 NEXT objekt
1735 FOR raum=1 TO ar
1736 FOR richtung=1 TO 6
1737 INPUT#9,durchgang(raum,richtung)
1738 NEXT richtung
1739 NEXT raum
1745 FOR flag=1 TO af
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1760 PRINT ok$
1770 GOTO 1080
1780 PRINT"Achtung Fehler !":BORDER 3:RE
SUME NEXT
1800 IF LEFT$(eingabe$,3)<>"VOK" THEN GO
TO 1900
1805 CLS:PRINT"Ich verstehe folgende Ver
ben:":PRINT:PRINT:RESTORE
1810 FOR i=1 TO av
1820 READ wort$:PRINT wort$
1830 NEXT i
1840 GOSUB 1890
1845 CLS:PRINT"und folgende Gegenstaende
 sind mir ":PRINT"bekannt:":PRINT
1849 zeile=0
1850 FOR i=1 TO ao
1855 zeile=zeile+1
1860 READ wort$,wort$,x:PRINT wort$
1865 IF zeile=20 THEN GOSUB 1890
1866 IF zeile=20 THEN zeile=1:CLS
```

```
1870 NEXT i
1880 GOSUB 1890:CLS:GOTO 1080
1890 LOCATE 1,25:PRINT "Taste druecken"
1895 eingabe$=INKEY$:IF eingabe$="" THEN
 1895 ELSE CLS:RETURN
1900 IF LEFT$(eingabe$,3)<>"INS" THEN GO
TO 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(eingabe$,3)<>"END" THEN GO
TO 1970 ELSE CLS
1960 PRINT"Der Autor wuenscht Ihnen mehr
 Erfolg":PRINT:PRINT"beim naechsten Mal
!":PRINT:PRINT:PRINT:END
1970 IF LEFT$(eingabe$,3)<>"INS" THEN GO
TO 2000
1971 IF spieler=4 AND ob(10)=0 THEN PRIN
T"Fast waere ich in eine Grube gefallen.
":GOTO 1080
1972 IF spieler=4 AND ob(11)<>spieler AN
D NOT fl(2) THEN PRINT"Ich brauche etwas
, um die Kette zu":PRINT"zerreissen !":G
OTO 1080
1975 PRINT"Erst sehen, dann denken !":PR
INT"Und zuletzt handeln.":GOTO 1080
1979 REM ******************** ENDE HELP
2000 laenge=LEN(eingabe$)
2010 FOR buchstabe=1 TO laenge
2020 pruef$=MID$(eingabe$,buchstabe,1)
2030 IF pruef$<>" "THEN NEXT buchstabe
2040 everb$=LEFT$(eingabe$,wortlaenge)
2050 rl=laenge-buchstabe
2060 IF rl<0 THEN 2090
2070 eobjekt$=RIGHT$(eingabe$,rl)
2080 eobjekt$=LEFT$(eobjekt$,wortlaenge)
2090 FOR verbnummer=1 TO av
2100 IF everb$=verb$(verbnummer) THEN 21
30
2110 NEXT verbnummer
2120 PRINT"Das Verb verstehe ich nicht !
":GOTO 1080
```

```
2130 FOR o=1 TO ao
2140 IF eobjekt$=rufname$(o) THEN 2200
2150 NEXT o
2160 PRINT"Ich verstehe das Objekt nicht
 !":GOTO 1080
2190 REM                    unte nimm leg
oeffne benutze zerstoere
2200 ON verbnummer GOTO 5000,2210,7000,8
000,9000,10000,11000,12000,13000,14000,1
5000
2209 REM ****************** INV BEGRENZEN
2210 anzahl=0:FOR i=1 TO ao
2220 IF ob(i)=-1 THEN anzahl=anzahl+1
2230 IF anzahl=imax THEN PRINT"Nein dank
e. - Ich trage schon genug.":GOTO 1080
2240 NEXT i
2250 GOTO 6000
2260 REM ****************** ende invtest
2999 REM UP Lichtschalter
3000 IF licht=-1 THEN licht=0:GOTO 3020
3010 IF licht=0 THEN licht=-1
3020 lw=0:RETURN
4500 CLS:REM                  SPIELER TOD
4600 PRINT"Auch das noch !":PRINT:PRINT
m$(0)
4610 PRINT:PRINT"Ich bin tod !":PRINT
4620 INPUT"Soll ich es noch einmal versu
chen ";eingabe$:eingabe$=UPPER$(eingabe$
)
4630 IF LEFT$(eingabe$,1)="J" THEN RUN
4640 GOTO 1960
4800 CLS:REM               SPIELER SIEGT
4810 PRINT"Herzlichen Glueckwunsch !"
4820 PRINT:PRINT"Sie haben die Ihnen ges
tellte Aufgabe":PRINT:PRINT"geloest und
duerfen sich nun an einem":PRINT:PRINT"a
nderen Adventure versuchen."
4830 PRINT:PRINT:PRINT:PRINT:PRINT:END
4999 REM *********** SPIELZUG AUSFUEHREN
5000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 5900
```

```
5002 IF o=1 THEN PRINT m$(1):GOTO 1080
5003 IF o=3 THEN PRINT m$(1):GOTO 1080
5004 IF o=4 THEN PRINT"In einer Ecke ste
ht ein Regal.":ob(8)=spieler:GOTO 1080
5005 IF o=5 THEN PRINT"Die Flasche ist g
efuellt mit Honig.":GOTO 1080
5006 IF o=6 THEN PRINT m$(1):GOTO 1080
5007 IF o=7 AND ob(9)=0 THEN PRINT"In de
r Kiste liegt Sprengstoff.":GOTO 1080
5008 IF o=8 AND ob(5)=0 THEN PRINT"Auf d
em Regal steht eine Korbflasche.":ob(5)=
spieler:GOTO 1080
5009 IF o=8 AND ob(5)<>0 THEN PRINT m$(1
):GOTO 1080
5010 IF o=10 THEN PRINT"In dem Erdloch l
iegt eine Eisentruhe.":ob(11)=spieler:GO
TO 1080
5011 IF o=11 AND NOT fl(1) THEN PRINT"Si
e ist mit einer Eisenkette ver-":PRINT"s
chlossen.":ob(19)=spieler:GOTO 1080
5012 IF o=11 AND fl(1) AND NOT fl(2) THE
N PRINT"Aussen sehe ich nichts besondere
s.":GOTO 1080
5013 IF o=11 AND fl(1) AND fl(2) AND ob(
12)=0 THEN PRINT"Sie ist voller Silbermu
enzen.":ob(12)=spieler:GOTO 1080
5014 IF o=12 THEN PRINT"Genau das suche
ich !":GOTO 1080
5015 IF o=13 THEN PRINT"Ich habe einen B
aeren aufgeschreckt !":ob(14)=spieler:fl
(3)=0:GOTO 1080
5017 IF o=15 THEN PRINT"Zwischen den Bue
schen ist ein Erdloch.":ob(10)=spieler:G
OTO 1080
5018 IF o=16 THEN PRINT"Sie sehen sehr s
tabil aus.":GOTO 1080
5019 IF o=17 THEN PRINT"Es handelt sich
um pures Gold !":GOTO 1080
5020 IF o=45 AND ob(21)=0 THEN PRINT"Dar
an haengt immer noch das Zugseil.":ob(21
)=spieler:GOTO 1080
```

```
5021 IF o=23 THEN PRINT"Es ist eine alte
 Oellampe.":GOTO 1080
5022 IF o=24 THEN PRINT"In der Wand stec
kt ein Haken.":ob(25)=spieler:GOTO 1080
5023 IF o=25 THEN PRINT"Tut mir leid - e
r steckt zu fest.":GOTO 1080
5024 IF o=26 THEN m$(0)="Er ist sehr tie
f - und ich war           zu dicht am Ran
d.":GOTO 4500
5025 IF o=27 THEN PRINT"Ihr Tischler war
 ein Koenner.":GOTO 1080
5026 IF o=28 AND ob(o)=0 THEN PRINT"Ich
habe zwei Saecke entdeckt.":ob(29)=spiel
er:GOTO 1080
5027 IF o=29 THEN PRINT"Sie sind gefuell
t mit Goldstaub !":GOTO 1080
5028 IF o=31 THEN PRINT"Sie ist oelig sc
hmierig und klebt an":PRINT"den Fingern.
":GOTO 1080
5029 IF o=32 THEN PRINT"Sie stinkt !"
5030 IF o=32 AND ob(34)=0 THEN PRINT"In
der Jackentasche sind Goldmuenzen.":ob(3
4)=spieler:GOTO 1080
5031 IF o=35 THEN PRINT"Es sind gar kein
e Spinnweben.":ob(o)=0:ob(42)=spieler:PR
INT"Es ist eine Zuendschnur.":GOTO 1080
5033 IF o=36 THEN PRINT"Es ist tatsaechl
ich reines Gold.":GOTO 1080
5034 IF o=37 THEN PRINT"Darauf steht:":P
RINT"Wenn erst mal die Gier erwacht":PRI
NT"der Tod auch oft Geschaefte macht.":G
OTO 1080
5036 IF o=42 THEN PRINT"Sie fuehrt hinau
f zur Decke.":GOTO 1080
5037 IF o=43 THEN PRINT"Es ist eins der
ueblichen Western-":PRINT"feuerzeuge.":G
OTO 1080
5038 IF o=32 AND fl(4)=-1 AND ob(33)<>-2
 THEN PRINT"Sie lag auf Silberstuecken."
:ob(33)=spieler:GOTO 1080
5800 PRINT m$(1):GOTO 1080
```

```
5900 REM      gegenstand nicht vorhanden
5901 IF o=1 AND spieler=2 THEN PRINT m$(
1):GOTO 1080
5902 IF o=6 AND ob(5)=-1 THEN PRINT"Er i
st suess und gut.":GOTO 1080
5903 IF o=6 AND ob(5)<>-1 THEN PRINT"Ich
 habe keinen Honig !":GOTO 1080
5904 IF o=9 AND (ob(7)=spieler OR ob(7)=
-1) THEN PRINT"Er sieht sehr explosiv au
s !":GOTO 1080
5905 IF o=16 AND ob(18)=-1 THEN PRINT"Ka
um angerostet - wirklich stabil !":GOTO
1080
5910 IF o=14 AND spieler=5 AND ob(14)=5
THEN PRINT"Ich scheine seinen Appetit an
zuregen.":GOTO 1080
5911 IF o=20 AND spieler=22 THEN PRINT"W
enn erst mal die Gier erwacht":PRINT"der
 Tod auch oft Geschaefte macht !":GOTO 1
080
5912 IF o=44 AND spieler=23 THEN m$(0)="
Ich bin im Totenreich.":GOTO 4500
5990 PRINT"So etwas sehe ich hier nicht
!":GOTO 1080
6000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 6900
6001 IF o=1 THEN PRINT m$(2):GOTO 1080
6002 IF o=3 THEN PRINT m$(2):GOTO 1080
6003 IF o=4 THEN PRINT m$(3):GOTO 1080
6004 IF o=8 THEN PRINT m$(2):GOTO 1080
6005 IF o=11 THEN PRINT m$(2):GOTO 1080
6006 IF o=10 THEN PRINT m$(3):GOTO 1080
6007 IF o=13 THEN PRINT m$(3):GOTO 1080
6008 IF o=15 THEN PRINT m$(2):GOTO 1080
6010 IF o=5 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
6011 IF o=6 THEN ob(5)=-1:PRINT ok$:GOTO
 1080
6012 IF o=7 THEN ob(o)=-1:PRINT ok$:GOTO
 1080
6014 IF o=12 THEN ob(o)=-2:wertung=wertu
```

```
ng+10:PRINT ok$:GOTO 1080
6015 IF o=14 AND spieler=5 THEN m$(0)="D
er Baer hat mich erschlagen.":GOTO 4500
6016 IF o=16 THEN ob(18)=-1:PRINT ok$:GO
TO 1080
6017 IF o=17 AND fl(3) THEN PRINT ok$:ob
(o)=-1:GOTO 1080
6018 IF o=17 AND NOT fl(3) THEN m$(0)="E
in Baer stuerzt sich auf mich.         E
r hat mich erschlagen.":GOTO 4500
6019 IF o=12 AND ob(o)=-2 THEN PRINT"Das
 Silber habe ich bereits.":GOTO 1080
6020 IF o=17 AND ob(o)=-2 THEN PRINT"Das
 Gold habe ich bereits.":GOTO 1080
6021 IF o=45 THEN PRINT m$(2):GOTO 1080
6022 IF o=21 AND fl(4)=-1 THEN PRINT ok$
:ob(o)=-1:GOTO 1080
6023 IF o=22 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6024 IF o=24 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6025 IF o=25 THEN PRINT"Er steckt zu tie
f im Fels.":GOTO 1080
6026 IF o=27 THEN PRINT m$(3):GOTO 1080
6027 IF o=28 THEN PRINT"Und was soll ich
 damit ?":GOTO 1080
6030 IF o=31 AND ob(5)<>-1 THEN PRINT"Wo
mit denn ?":GOTO 1080
6031 IF o=31 AND ob(5)=-1 THEN PRINT ok$
;" - die Flasche ist voll.":fl(6)=-1:GOT
O 1080
6032 IF o=32 THEN PRINT ok$;" - oh was i
st das denn ?":fl(5)=-1:GOTO 1080
6033 IF o=33 AND ob(o)=spieler THEN PRIN
T ok$:ob(o)=-2:wertung=wertung+10:GOTO 1
080
6034 IF o=34 AND ob(o)=spieler THEN PRIN
T ok$:ob(o)=-2:wertung=wertung+10:GOTO 1
080
6035 IF o=35 THEN PRINT"Es sind gar kein
e Spinnweben, es war":PRINT"eine Zuendsc
```

```
hnur.":ob(o)=0:ob(42)=-1:GOTO 1080
6037 IF o=37 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6038 IF o=41 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6039 IF o=42 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6040 IF o=43 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6041 IF o=21 AND NOT fl(4) THEN PRINT"Es
 ist an der Lore befestigt.":GOTO 1080
6042 IF o=23 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6043 IF o=29 AND ob(o)=spieler THEN PRIN
T ok$:ob(o)=-2:wertung=wertung+10:GOTO 1
080
6044 IF o=38 AND ob(o)=spieler THEN PRIN
T ok$:ob(o)=-2:wertung=wertung+10:GOTO 1
080
6900 IF o=9 AND (ob(7)=spieler OR ob(7)=
-1) THEN m$(0)="Bei der Beruehrung ist d
er Sprengstoff  explodiert !":GOTO 4500
6901 IF o=16 AND ob(18)=spieler THEN PRI
NT ok$:ob(18)=-1:GOTO 1080
6910 IF o=1 AND spieler=2 THEN PRINT m$(
2):GOTO 1080
6911 IF o=20 AND spieler=22 THEN PRINT"W
enn erst mal die Gier erwacht":PRINT"der
 Tod auch oft Geschaefte macht.":GOTO 10
80
6999 PRINT"So etwas sehe ich hier nicht
!":GOTO 1080
7000 IF o=16 AND ob(18)=-1 THEN PRINT ok
$:ob(18)=spieler:GOTO 1080
7001 IF ob(o)<>-1 THEN PRINT"So etwas be
sitze ich doch gar nicht !":GOTO 1080
7010 IF o=6 AND spieler=5 THEN ob(6)=0:f
l(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0
:GOTO 1080
7020 IF o=5 AND spieler=5 THEN ob(5)=0:f
l(3)=-1:PRINT m$(4):PRINT m$(5):ob(14)=0
```

```
:GOTO 1080
7900 ob(o)=spieler:PRINT ok$:GOTO 1080
8000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N PRINT"So etwas ist hier nicht !":GOTO
1080
8005 IF o=4 AND spieler=3 THEN PRINT"Die
 Huette war bereits offen.":GOTO 1080
8010 IF o=5 THEN PRINT ok$:GOTO 1080
8020 IF o=11 AND NOT fl(1) THEN PRINT"Da
s laesst die Kette nicht zu.":GOTO 1080
8025 IF o=11 AND fl(1) THEN PRINT ok$;"
- der Deckel klappt nach hinten.":fl(2)=
-1:GOTO 1080
8030 IF o=23 THEN PRINT ok$:GOTO 1080
8035 IF o=29 THEN PRINT ok$:GOTO 1080
8040 IF o=32 THEN PRINT"Tut mir leid - i
ch bin nicht Frankenstein !":GOTO 1080
8045 IF o=36 THEN PRINT"Wie ?":GOTO 1080
8999 PRINT"Ich verstehe nicht, was Du me
inst.":GOTO 1080
9000 IF o=16 AND ob(18)=-1 AND spieler=4
 THEN PRINT"Die Kette zerspringt.":fl(1)
=-1:GOTO 1080
9005 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 9900
9010 IF o=16 AND spieler=4 THEN PRINT"Di
e Kette zerspringt.":fl(1)=-1:GOTO 1080
9020 IF o=45 THEN PRINT"Wie und wozu ?":
GOTO 1080
9030 IF o=21 THEN PRINT"Wie und wozu ?":
GOTO 1080
9999 PRINT"Ich verstehe nicht, was Du me
inst.":GOTO 1080
10000 IF o=16 AND spieler=4 AND ob(18)=-
1 THEN PRINT"Die Kette zerspringt.":fl(1
)=-1:GOTO 1080
10010 IF o=16 AND spieler=4 AND ob(18)<>
-1 THEN PRINT"Womit ?":GOTO 1080
10020 IF o=36 AND ob(22)=-1 THEN PRINT o
k$:durchgang(22,1)=23:GOTO 1080
10030 IF o=21 AND (ob(o)=spieler OR ob(o
```

```
)=-1) THEN PRINT ok$:fl(4)=-1:GOTO 1080
10040 IF o=27 AND ob(22)=-1 THEN PRINT o
k$:durchgang(10,2)=12:GOTO 1080
10050 IF o=27 AND ob(22)<>-1 THEN PRINT"
Womit ?":GOTO 1080
10999 PRINT"Ich verstehe nicht, was Du m
einst !":GOTO 1080
11000 IF ob(43)<>-1 THEN PRINT"Ich habe
nichts zum Zuenden.":GOTO 1080
11010 IF o=8 AND lz<=0 THEN PRINT"In der
 Lampe ist kein Oel mehr !":GOTO 1080
11020 IF o=42 AND (ob(43)=-1 OR licht=-1
) THEN PRINT"zzzisch !":fl(7)=-1:ex=zug+
3:ob(o)=0:durchgang(30,2)=31:durchgang(3
1,1)=30:GOTO 1080
11030 IF o=23 AND ob(23)=-1 THEN PRINT o
k$;" - die Lampe brennt.":lm=11:lw=1:GOT
O 1080
11040 IF o=43 THEN PRINT ok$;" - der Doc
ht glimmt.":GOTO 1080
11098 PRINT"Ich verstehe Dich nicht.":GO
TO 1080
12000 IF o=23 AND (fl(6) AND ob(23)=-1)
THEN lm=60:fl(6)=0:lw=0:PRINT ok$:GOTO 1
080
12020 IF o=5 AND spieler=17 THEN fl(6)=-
1:PRINT"Die Flasche ist voll.":GOTO 1080
12030 IF o=43 AND ob(o)=-1 THEN PRINT"De
r Docht glimmt.":GOTO 1080
12998 PRINT"Ich verstehe nicht, was Du m
einst.":GOTO 1080
13000 IF o=13 AND spieler=5 THEN spieler
=12:PRINT ok$:GOTO 1080
13998 PRINT"Ich weiss nicht, was Du will
st.":GOTO 1080
14000 IF o=23 AND ob(o)=-1 THEN ll=lm-lw
:PRINT ok$:GOTO 1080
14998 PRINT"Ich weiss nicht, was Du will
st.":GOTO 1080
15000 IF o=21 AND spieler=9 THEN PRINT o
k$:ob(o)=9:durchgang(9,6)=18:durchgang(1
```

```
8,5)=9:GOTO 1080
15998 PRINT"Ich weiss nicht, was Du will
st.":GOTO 1080
```

SPACE MISSION

Bitte lassen Sie sich durch den Reiz eines Grafikadventures nicht dazu verleiten, dieses Programm einzugeben, sofern Sie absoluter Newcomer auf diesem Gebiet sind und Ihre Programmierkenntnisse darüber hinaus nicht so weit fortgeschritten sein sollten, daß Sie sich die Antworten auf die sich Ihnen sehr schnell stellenden Probleme aus dem Listing holen können.

Denn im Gegensatz zu den übrigen Adventurespielen, bei denen gefährliche Situationen erst durch Ihr Walten und Schalten entstehen, haben Sie bei Space Mission nach der Eingabe von RUN nur neun (9!) Spielzüge Zeit, um Ihr Leben zu retten.

Und es wäre doch schade, wenn Sie die Lust an Spielprogrammen dieser Art verlieren, nur weil Sie kein Erfolgserlebnis haben !

```
1 REM space mission
2 REM (c) 1984 by Walkowiak
3 REM ********************************
10 CLS:LOCATE 14,22:PRINT"SPACE MISSION"
20 LOCATE 10,23:PRINT"(c) 1984 by Walkow
iak"
30 FOR i=1 TO 8000:NEXT
150  ar= 30:ao= 50:av= 10:af= 11:imax=2:
zug=0
160  spieler= 11
170  wl= 4
190  DIM raum$(ar),durchgang(ar,6),ob$(a
o),rufname$(ao),ob(ao),verb$(av),fl(af)
200  REM ************************* VERBE
N
210  DATA untersuche
211  DATA nimm
212  DATA oeffne
213  DATA druecke
214 DATA "leg "
215  DATA trage
216  DATA fuelle
217  DATA setze
218  DATA benutze
219  DATA tausche
300  REM ******************* GEGENSTAEND
E
301  DATA das Zentralschaltpult,Pult,1
302  DATA einen Videoschirm,Videoschirm,
1
303  DATA ,Schalter,0
304  DATA ein Terminal,Terminal,12
305  DATA einen Hyperkom,Hyperkom,0
306  DATA einen Telekom,Telekom,0
307 DATA ,Antigravschacht,0
308  DATA einen Einstieg,Einstieg,0
309  DATA einen Einstieg,Einstieg,0
310  DATA ein riesiges Blechgehaeuse,Geh
aeuse,7
311  DATA den Zentralcomputer,Computer,0
312  DATA ein Steckmodul,Steckmodul,-1
```

```
313  DATA Module,Module,0
314  DATA einen Inbusschluessel,Inbus,0
315  DATA Kabelbaeume,Kabel,7
316  DATA Rohre,Rohre,21
317  DATA ein Hinweisschild,Schild,8
318  DATA ,LAND,0
319  DATA einen Wandschrank,Wandschrank,
10
320  DATA ,Karte,0
321  DATA eine Tuer,Tuer,9
322  DATA den Autopiloten,Autopilot,0
323  DATA einen Schaltknopf,Knopf,9
324  DATA eine Pritsche,Pritsche,10
325  DATA nichts besonderes,Tisch,11
326  DATA ,MOVE,0
327  DATA mehrere Tanks,Tank,21
328  DATA eine Sauerstoffflasche,Sauerst
offflasche,20
329  DATA einen Raumanzug,Raumanzug,0
330  DATA einen 10'er Schluessel,zehner,
0
331  DATA ,STOP,0
332  DATA einen Schraubendreher,Schraube
ndreher,0
333  DATA den Leitstand,Leitstand,14
334  DATA eine rote Tuer,Tuer,14
335  DATA ,SETC,0
336  DATA ein Modul,Modul,0
337  DATA Medikamente,Medikamente,0
338  DATA ,Koordinaten,0
339  DATA Funkgeraete,Funkgeraete,3
340  DATA eine Magnetkarte,Magnetkarte,0
341  DATA nichts besonderes,Schott,16
342  DATA ,*SD*,0
343  DATA ,VIEW,0
344  DATA "nichts besonderes",Bett,30
345  DATA ,STAT,0
346  DATA ,SELO,0
347  DATA ,Transmitter,0
348  DATA nichts besonderes,Kontrolle,6
349  DATA ,Password,0
```

```
350 DATA Paeckchen,Paeckchen,15
500  REM ************* RAUMBESCHREIBUNGE
N
501  DATA in der Hauptzentrale,3,3,16,0,
0,0
502  DATA im Transmitterraum,6,0,0,0,0,0
503  DATA in der Funkzentrale,1,1,0,0,0,
0
504  DATA am pilotenpult,3,0,0,6,0,0
505  DATA ,0,0,0,0,0,0
506  DATA vor einer Zugangskontrolle,0,2
,0,0,0,0
507  DATA in einem Zwischendeck,0,0,23,0
,0,0
508  DATA in der astronomischen Abt.,0,0
,22,0,0,0
509  DATA in der oberen Polkuppel,0,0,0,
0,0,22
510  DATA in der Medostation,0,0,0,22,0,
0
511  DATA in der Kantine,0,0,30,24,0,0
512  DATA im Transmitterraum,12,12,18,13
,0,0
513  DATA in einem Frachtraum,0,0,12,0,0
,0
514  DATA im Maschinenraum,0,0,17,0,0,0
515  DATA in der Lagerhalle,0,2,0,0,0,0
516  DATA vor einem Sicherheitsschott,0,
0,24,0,0,0
517  DATA auf einem Gang,19,19,26,14,0,0
518  DATA auf einem Gang,18,18,27,12,0,0
519  DATA auf einem Gang,17,17,20,0,0,0
520  DATA im Ersatzteillager,0,0,0,19,0,
0
521  DATA in einem Zwischendeck,0,0,25,0
,0,0
522  DATA im Antigravschacht,0,0,10,8,9,
23
523  DATA im Antigravschacht,0,0,0,0,22,
24
524  DATA im Antigravschacht,0,0,11,16,2
```

```
3,0
525  DATA im Antigravschacht,0,0,0,0,24,
26
526  DATA im Antigravschacht,0,0,0,17,25
,27
527  DATA im Antigravschacht,0,0,0,18,26
,0
528 DATA ,,,,,,,,,,,,,
530  DATA in einem Mannschaftsraum,0,0,0
,11,0,0
600  REM ******************** MITTEILUNGE
N
601  m$(1)="Ich sehe nichts besonderes."
602  m$(2)="So stark bin ich nicht !"
603  m$(3)="Wie stellst Du Dir das vor ?
"
604 m$(4)="Das habe ich doch schon !"
605  m$(5)="Ich verstehe nicht, was Du m
einst !"
690 ok$="Okay !"
699  REM *********** hier evtl. 2. Titel
800  FOR i=1 TO av
810  READ verb$(i):verb$(i)=LEFT$(verb$(
i),wl):verb$(i)=UPPER$(verb$(i))
820  NEXT i
830  FOR objekt=1 TO ao
840  READ ob$(objekt),rufname$(objekt),o
b(objekt):rufname$(objekt)=LEFT$(rufname
$(objekt),wl):rufname$(objekt)=UPPER$(ru
fname$(objekt)):NEXT objekt
850  FOR raum=1 TO ar:READ raum$(raum)
860  FOR richtung=1 TO 6:READ durchgang(
raum,richtung)
870  NEXT richtung:NEXT raum
880  PRINT:INPUT"   Wuenschen Sie Ratsch
laege fuer Ihr                weiteres Vo
rgehen";eingabe$:eingabe$=UPPER$(eingabe
$)
890  IF LEFT$(eingabe$,1)="J"THEN GOSUB
900:GOTO 1000 ELSE GOTO 1000
900  MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
```

```
 1:INK 1,25
910 alt=0: PRINT"       CPC - Ventures":P
RINT TAB(15)CHR$(164);" 1984  by J"CHR$(
178);"rg Walkowiak
920  PRINT STRING$(40,208):PRINT"Stellen
 Sie  sich einen Roboter vor, denSie  mi
t zahlreichen  Kommandos  steuernkoennen
."
930  PRINT"Ich bin dieser  Roboter,  und
 ich  werdemich fuer Sie den Gefahren de
r verwegen-sten Abenteuer aussetzen."
940  PRINT"Damit Sie  mich sinvoll  agie
ren  lassenkoennen,  werde ich  Ihnen di
e Situationin der ich  mich gerade befin
de, jeweils genau beschreiben."
950  PRINT"Anschliessend  sagen  Sie  mi
r  mit zweiWorten,  wie  beispielsweise
 UNTERSUCHETUER, NIMM MESSER, was ich tu
n soll.":PRINT
960  PRINT"Darueber hinaus verstehe ich
die Befehle      INVENTUR, SAVE, LOAD und
 ENDE."
970  PRINT STRING$(40,210):INK 3,12,24:I
NK 2,24,12:PEN 3: PRINT"            <Tas
te>";:PEN 2:PRINT" druecken";:PEN 1:PRIN
T
980  eingabe$=INKEY$:IF eingabe$="" THEN
 980 ELSE MODE 1:INK 1,26:INK 2,1:INK 3,
0:RETURN
1000 INK 1,26:INK 2,1:INK 3,0:leerzeile$
=STRING$(40,32)
1010  DATA Norden,Sueden,Westen,Osten,ob
en,unten
1020  FOR richtung=1 TO 6
1030  READ richtung$(richtung)
1040  NEXT richtung
1050 WINDOW#1,1,40,1,16
1070 WINDOW#0,1,40,17,25:PAPER 2:INK 2,1
4:CLS#0
1080  PRINT:zug=zug+1:IF alt<>spieler TH
EN GOSUB 30000
```

```
1090  LOCATE 1,1
1100  FOR zeile=1 TO 7
1110  PRINT leerzeile$;
1120  NEXT zeile
1130  LOCATE 1,1:PEN 1
1140  PRINT"Ich bin ";:alt=spieler
1150  PRINT raum$(spieler);:PRINT"."
1160  PRINT"Ich sehe ";:gedruckt=0
1170  FOR i=1 TO ao
1180  IF ob(i)<>spieler THEN 1210
1190  IF POS(#0)+LEN(ob$(i))+2<=39 THEN
PRINT ob$(i);", ";:GOTO 1205
1200  IF POS(#0)+LEN(ob$(i))+2>39 THEN P
RINT:GOTO 1190
1205  gedruckt=-1
1210  NEXT i
1215  IF NOT gedruckt THEN PRINT"nichts
besonderes  ";
1220  PRINT CHR$(8);CHR$(8);"."
1230  PRINT leerzeile$
1240  PRINT"Ich kann nach ";:gedruckt=0
1250  FOR richtung=1 TO 6
1260  IF durchgang(spieler,richtung)=0 T
HEN GOTO 1300 ELSE gedruckt=-1
1270  IF POS(#0)=15 THEN PRINT richtung$
(richtung);:GOTO 1300
1280  IF POS(#0)+LEN(richtung$(richtung)
)<38 THEN PRINT", ";richtung$(richtung);
:GOTO 1300
1290  PRINT",":PRINT richtung$(richtung)
;:GOTO 1300
1300  NEXT richtung
1310  IF gedruckt=0 THEN PRINT"nirgendwo
";
1320  PRINT".":PEN 3
1350 IF fl(1)=0 THEN PRINT"Alarm !";:IF
zug>11 THEN m$(0)="Ein Asteroid hat uns
gerammt.":GOTO 4500
1390  PEN 3:LOCATE 1,9:INPUT"Was soll ic
h tun";eingabe$:eingabe$=UPPER$(eingabe$
)
```

```
1400  IF LEN(eingabe$)>2 THEN 1500
1410  IF eingabe$="N" AND durchgang(spie
ler,1)<>0 THEN spieler=durchgang(spieler
,1):PRINT ok$:GOTO 1080
1420  IF eingabe$="S" AND durchgang(spie
ler,2)<>0 THEN spieler=durchgang(spieler
,2):PRINT ok$:GOTO 1080
1430  IF eingabe$="W" AND durchgang(spie
ler,3)<>0 THEN spieler=durchgang(spieler
,3):PRINT ok$:GOTO 1080
1440  IF eingabe$="O" AND durchgang(spie
ler,4)<>0 THEN spieler=durchgang(spieler
,4):PRINT ok$:GOTO 1080
1450  IF eingabe$="OB" AND durchgang(spi
eler,5)<>0 THEN spieler=durchgang(spiele
r,5):PRINT ok$:GOTO 1080
1460  IF eingabe$="U" AND durchgang(spie
ler,6)<>0 THEN spieler=durchgang(spieler
,6):PRINT ok$:GOTO 1080
1470  IF LEN(eingabe$)<3 THEN PRINT"Dahi
n fuehrt kein Weg !":GOTO 1080
1480  IF LEN(eingabe$)>8 THEN GOTO 2000
1499  REM **************** START INVENTU
R
1500  IF LEFT$(eingabe$,3)<>"INV" THEN G
OTO 1600
1510  PRINT"Ich trage folgendes mit mir:
"
1520  FOR objekt=1 TO ao
1530  IF ob(objekt)=-1 THEN PRINT ob$(ob
jekt)
1540  NEXT objekt
1550  GOTO 1080
1560  REM **************** ENDE INVENTUR
1600  IF LEFT$(eingabe$,3)<>"SAV" THEN G
OTO 1700
1610  PRINT"REC & PLAY druecken ...
              Unter welchem Namen speicher
n ..........";STRING$(10,8);:INPUT einga
be$
1620  IF LEN(eingabe$)>10 THEN PRINT"Bit
```

```
te etwas kuerzer !":GOTO 1610 ELSE einga
be$="!"+eingabe$+".spiel":OPENOUT eingab
e$
1630  PRINT#9,spieler
1640  FOR objekt=1 TO ao
1650  PRINT#9,ob(objekt)
1660  NEXT objekt
1670  FOR raum=1 TO ar:FOR richtung=1 TO
 6:PRINT#9,durchgang(raum,richtung):NEXT
 richtung:NEXT raum
1680  FOR flag=1 TO af:PRINT#9,fl(flag):
NEXT flag
1690  CLOSEOUT:PRINT ok$:GOTO 1080
1700  IF LEFT$(eingabe$,3)<>"LOA" THEN G
OTO 1900
1710  PRINT"Cassette rueckspulen & PLAY
druecken ...Welches Spiel laden ........
..";STRING$(10,8);:INPUT eingabe$
1720  IF LEN(eingabe$)>10 THEN PRINT"Das
 kann nicht sein !":GOTO 1710 ELSE einga
be$="!"+eingabe$+".SPIEL":OPENIN eingabe
$
1730  INPUT#9,spieler
1740  FOR objekt=1 TO ao:INPUT#9,ob(obje
kt):NEXT objekt
1750  FOR raum=1 TO ar:FOR richtung=1 TO
 6:INPUT#9,durchgang(raum,richtung):NEXT
 richtung:NEXT raum
1760  FOR flag=1 TO af:INPUT#9,fl(flag):
NEXT flag
1770  CLOSEIN:PRINT ok$:GOTO 1080
1900  IF LEFT$(eingabe$,3)<>"INS" THEN G
OTO 1950
1910  GOSUB 900:GOTO 1070
1950  IF LEFT$(eingabe$,3)<>"END" THEN G
OTO 2000
1960 MODE 1:PRINT"Der Autor wuenscht Ihn
en mehr Erfolg    beim naechsten Mal.":P
RINT:PRINT:PRINT:END
2000  laenge=LEN(eingabe$)
2010  FOR buchstabe=1 TO laenge
```

```
2020  pruef$=MID$(eingabe$,buchstabe,1)
2030  IF pruef$<>" "THEN NEXT buchstabe
2040  everb$=LEFT$(eingabe$,wl)
2050  rl=laenge-buchstabe
2060  IF rl<0 THEN 2090
2070  eobjekt$=RIGHT$(eingabe$,rl)
2080  eobjekt$=LEFT$(eobjekt$,wl)
2090  FOR verbnummer=1 TO av
2100  IF everb$=verb$(verbnummer) THEN 2
130
2110  NEXT verbnummer
2120  PRINT"Das Verb verstehe ich nicht.
":GOTO 1080
2130  FOR o=1 TO ao
2140  IF eobjekt$=rufname$(o) THEN 2200
2150  NEXT o
2160  PRINT"Diesen Gegenstand kenne ich
nicht.":GOTO 1080
2200  ON verbnummer GOTO 5000, 2210, 700
0, 8000, 9000, 10000, 11000, 12000, 1300
0, 14000
2210 anzahl=0:FOR i=1 TO ao
2220 IF ob(i)=-1 THEN anzahl=anzahl+1
2230 IF anzahl=imax THEN PRINT"Nein dank
e. - Ich trage schon genug.":GOTO 1080
2240 NEXT i
2250 GOTO 6000
4500 MODE 1:CLS:REM SPIELER TOD
4510  PRINT"Auch das noch !":PRINT:PRINT
 m$(0)
4520  PRINT:PRINT"Ich bin tod !":PRINT
4530  INPUT"Soll ich es noch einmal vers
uchen ";eingabe$:eingabe$=UPPER$(eingabe
$)
4540  IF LEFT$(eingabe$,1)="J" THEN RUN
4550  GOTO 1960
4800 MODE 1:REM SPIELER SIEGT
4810  PRINT"Herzlichen Glueckwunsch !"
4815 PRINT:PRINT"Sie haben Pia 3 rechtze
itig erreicht,"
4820  PRINT:PRINT"und die Ihnen gestellt
```

```
e Aufgabe somit":PRINT:PRINT"geloest. Si
e duerfen sich nun an einem":PRINT:PRINT
"anderen Adventure versuchen."
4830  PRINT:PRINT:PRINT:PRINT:END
5000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 5900
5005 IF o=1 THEN PRINT"Damit bediene ich
 u.a. den Autopiloten.":ob(22)=1:GOTO 10
80
5006 IF o=2 AND fl(1)=0 THEN PRINT"Der S
chirm zeigt einen Asteroiden.":GOTO 1080
5007 IF o=2 AND fl(1)=-1 THEN PRINT"Ich
sehe nichts besonderes.":GOTO 1080
5009 IF o=40 THEN GOTO 51500
5010 IF o=6 THEN PRINT m$(1):GOTO 1080
5013 IF o=25 AND ob(40)=0 THEN PRINT"Dar
unter lag eine Magnetkarte.":ob(40)=11:a
lt=0:GOSUB 30000:GOTO 1080
5014 IF o=8 AND spieler=23 AND fl(7)=0 T
HEN PRINT"Er ist verschlossen.":GOTO 108
0
5015 IF o=8 AND spieler=25 AND fl(8)=0 T
HEN PRINT"Er ist verschlossen.":GOTO 108
0
5016 IF o=8 AND spieler=23 AND fl(7)=-1
THEN PRINT"Er fuehrt in ein Zwischendeck
.":GOTO 1080
5017 IF o=8 AND spieler=25 AND fl(8)=-1
THEN PRINT"Er fuehrt in ein Zwischendeck
.":GOTO 1080
5019 IF o=39 THEN PRINT m$(1):ob(5)=spie
ler:ob(6)=spieler:ob(39)=0:GOTO 1080
5020 IF o=5 THEN GOTO 20000
5021 IF o=44 AND ob(29)=0 THEN PRINT"Dar
unter liegt mein Raumanzug.":GOTO 1080
5025 IF o=4 THEN PRINT"Die Ziffern von 0
 bis 9 sind vorhanden.":GOTO 1080
5026 IF o=10 AND ob(32)<>-1 THEN PRINT"I
ch kann es nicht oeffnen.":GOTO 1080
5027 IF o=10 AND ob(32)=-1 THEN PRINT"Da
rin ist der Zentralrechner.":ob(11)=7:ob
```

```
(10)=0:GOTO 1080
5028 IF o=11 THEN PRINT"Ich sehe zahlrei
che Module.":ob(13)=spieler:ob(15)=0:GOT
O 1080
5029 IF o=13 THEN PRINT"Es sind Steckmod
ule.":GOTO 1080
5030 IF o=12 THEN PRINT"... ein normales
 Computermodul.":GOTO 1080
5031 IF o=14 THEN PRINT m$(1):GOTO 1080
5032 IF o=15 THEN PRINT m$(1):GOTO 1080
5033 IF o=16 THEN PRINT"Sie gehoeren zum
 Belueftungssystem.":GOTO 1080
5034 IF o=17 THEN PRINT"Nur ein Wegweise
r.":GOTO 1080
5035 IF o=20 THEN PRINT m$(1):GOTO 1080
5036 IF o=33 THEN PRINT m$(1):ob(30)=spi
eler:GOTO 1080
5037 IF o=22 THEN PRINT"Ich sehe Tasten:
 LAND MOVE STOP & SETC.":GOTO 1080
5038 IF o=23 THEN PRINT m$(1):GOTO 1080
5039 IF o=24 AND ob(32)=0 THEN PRINT"Dah
inter liegt ein Schraubendreher.":GOTO 1
080
5040 IF o=32 THEN PRINT m$(1):GOTO 1080
5041 IF o=27 THEN PRINT m$(1):GOTO 1080
5042 IF o=28 AND fl(4)=0 THEN PRINT"Sie
ist leer.":GOTO 1080
5043 IF o=28 AND fl(4)=-1 THEN PRINT"Sie
 ist gefuellt.":GOTO 1080
5044 IF o=29 THEN PRINT"Es ist meiner.":
GOTO 1080
5045 IF o=30 THEN PRINT m$(1):GOTO 1080
5046 IF o=48 THEN PRINT m$(1):GOTO 1080
5200 IF o=24 THEN PRINT m$(1):GOTO 1080
5899 PRINT m$(1):GOTO 1080
5900 IF o=7 AND spieler=24 THEN PRINT m$
(1):GOTO 1080
5902 IF o=7 AND spieler=23 THEN fl(5)=-1
:alt=0:ob(8)=spieler:GOTO 1080
5903 IF o=7 AND spieler=24 THEN PRINT m$
(1):GOTO 1080
```

```
5904 IF o=7 AND spieler=25 THEN fl(6)=-1
:ob(9)=spieler:alt=1:GOTO 1080
5905 IF o=7 AND spieler=26 THEN PRINT m$
(1):GOTO 1080
5906 IF o=7 AND spieler=27 THEN PRINT m$
(1):GOTO 1080
5907 IF o=7 AND spieler=22 THEN PRINT m$
(1):GOTO 1080
5910 IF o=20 AND spieler=8 THEN PRINT"PI
A III ist mit 151064 angegeben.":GOTO 10
80
5916 IF o=25 AND ob(40)=-1 THEN GOTO 108
0
5920 IF o=32 THEN PRINT m$(1):GOTO 1080
5922 IF o=8 AND spieler=25 AND durchgang
(25,4)=0 THEN PRINT"Er ist verschlossen.
":GOTO 1080
5923 IF o=47 AND spieler=2 THEN PRINT"Da
s gleiche Modell.":GOTO 1080
5924 IF o=4 AND spieler=2 THEN PRINT"Das
 gleiche Modell.":GOTO 1080
5925 IF o=20 AND spieler=15 THEN PRINT"E
s sind die Medikamente.":GOTO 1080
5990 PRINT"So etwas sehe ich hier nicht.
":GOTO 1080
6000 IF ob(o)<>spieler AND ob(o)<>-1 THE
N GOTO 6900
6005 IF o=40 THEN PRINT ok$:ob(o)=-1:alt
=0:GOSUB 30000:GOTO 1080
6010 IF o=12 AND ob(o)=-1 THEN PRINT m$(
4):GOTO 1080
6011 IF o=12 THEN ob(12)=-1:PRINT ok$:GO
TO 1080
6012 IF o=13 THEN PRINT"Ich habe ein Mod
ul.":ob(36)=-1:GOTO 1080
6013 IF o=14 OR o=28 THEN PRINT ok$:ob(o
)=-1:GOTO 1080
6014 IF (o=30 OR o=50) THEN PRINT ok$:ob
(o)=-1:alt=0:GOTO 1080
6800 IF o=32 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
```

```
6890 IF o=spieler THEN PRINT m$(2):GOTO
1080
6900 IF o=29 THEN PRINT ok$:ob(o)=-1:GOT
O 1080
6915 IF o=20 AND spieler=15 THEN PRINT o
k$:ob(50)=-1:GOTO 1080
6920 IF o=32 AND ob(o)=0 THEN ob(o)=-1:P
RINT ok$:GOTO 1080
6990 PRINT"So etwas sehe ich hier nicht.
":GOTO 1080
7000 IF o=41 AND ob(40)=-1 THEN PRINT ok
$:durchgang(16,4)=1:alt=2:GOTO 1080
7005 IF o=41  AND ob(40)<>-1 THEN PRINT"
Das geht im Moment nicht.":GOTO 1080
7010 IF o=8 AND ob(14)<>-1 THEN PRINT"Ic
h habe nichts zum oeffnen.":GOTO 1080
7011 IF o=8 AND spieler=23 AND ob(14)=-1
 THEN PRINT ok$:durchgang(23,4)=7:fl(7)=
-1:GOTO 1080
7012 IF o=8 AND spieler=25 AND ob(14)=-1
 THEN PRINT ok$:durchgang(25,4)=21:fl(8)
=-1:GOTO 1080
7015 IF o=23 AND spieler=9 THEN m$(0)="E
ntweichender Sauerstoff entreisst mich i
ns All.":GOTO 4500
7020 IF o=9 AND spieler=14 THEN m$(0)="E
ine Funkenentladung aus dem Reaktorraumt
raf mich.":GOTO 4500
7099 PRINT m$(5):GOTO 1080
8000 IF o=46 AND fl(2)=-1 THEN PRINT"Kli
ck !":durchgang(24,6)=25:GOTO 1080
8012 IF o=46 AND fl(2)=0 THEN PRINT"Nich
ts geschieht.":GOTO 1080
8020 IF o=45 THEN GOTO 20200
8021 IF o=43 THEN GOTO 20300
8022 IF o=42 THEN m$(0)="Das war die Sel
bstzertstoerung.":GOTO 4500
8024 IF o=18 THEN m$(0)="Der Autopilot l
andete das Schiff in der Sonne.":GOTO 45
00
8025 IF o=26 AND tm$="151063" THEN m$(0)
```

```
="Die Fehlfunktion - falsche Koordinaten
.":GOTO 4500
8026 IF o=26 AND fl(1)=0 THEN PRINT"Das
Schiff weicht dem Asteroiden aus.":fl(1)
=-1:GOTO 1080
8027 IF o=26 AND fl(1)=-1 AND fl(10)=0 T
HEN m$(0)="Keine Kursdaten gesetzt - das
 Ziel war  eine Sonne.":GOTO 4500
8028 IF o=26 AND fl(10)=-1 AND ob(50)<>1
3 THEN m$(0)="Die Bewohner von PIA III h
aben mich      gelyncht, weil ich ohne di
e Medikamente gekommen bin.":GOTO 4500
8029 IF o=31 THEN fl(1)=0:PRINT ok$:GOTO
 1080
8030 IF o=35 THEN INPUT"Koordinaten ";tm
$:IF tm$="151064" THEN fl(10)=-1:GOTO 10
80
8031 IF o=35 THEN GOTO 1080
8032 IF o=26 AND fl(10)=-1 AND ob(50)=13
 THEN GOTO 4800
8115 IF o=23 AND spieler=9 THEN m$(0)="E
ntweichender Sauerstoff riss mich ins  A
ll.":GOTO 4500
8990 PRINT"So einen Knopf sehe ich hier
nicht.":GOTO 1080
9000 IF ob(o)<>-1 THEN GOTO 9900
9005 IF o=50 AND ob(50)=-1 AND spieler=1
3 THEN PRINT ok$:ob(50)=spieler:alt=0:fl
(11)=-1:GOTO 1080
9006 IF o=50 AND ob(50)=-1 AND spieler=1
5 THEN PRINT ok$:ob(50)=spieler:alt=0:fl
(11)=-1:GOTO 1080
9010 PRINT ok$:ob(o)=spieler:GOTO 1080
9900 IF o=13 AND ob(36)=-1 THEN ob(36)=s
pieler:PRINT ok$:GOTO 1080
10000 IF o=29 THEN PRINT"Ich habe den Ra
umanzug angelegt.":ob(o)=-2:ob(14)=-1:GO
TO 1080
10010 GOTO 6000
11000 IF o=28 AND spieler<>21 THEN PRINT
"Womit ?":GOTO 1080
```

```
11010 IF o=28 AND spieler=21 AND ob(30)<
>-1 THEN PRINT"Das geht im Moment nicht.
":GOTO 1080
11020 IF o=28 AND spieler=21 AND ob(30)=
-1 THEN PRINT ok$:fl(4)=-1:GOTO 1080
11900 PRINT m$(5):GOTO 1080
12000 IF o=38 AND spieler=12 THEN INPUT
tm$:GOTO 1080
12005 IF o=38 AND spieler=2 THEN INPUT t
m$:GOTO 1080
12010 IF o=49 AND spieler=6 THEN INPUT t
m$:IF tm$="220559" THEN durchgang(2,1)=1
5:spieler=15:GOTO 1080
12020 IF o=49 AND spieler=6 AND tm$<>"22
0559" THEN m$(0)="Die Abwehrssysteme spr
echen an.":GOTO 4500
13000 IF o=47 AND NOT(spieler=12 OR spie
ler=2) THEN PRINT"Ich sehe keinen Transm
itter.":GOTO 1080
13010 IF o=47 AND spieler=12 AND tm$<>"6
44664" THEN m$(0)="Meine Atome treiben i
m Hyperraum.":GOTO 4500
13011 IF o=47 AND spieler=2 AND tm$<>"64
4663" THEN m$(0)="Meine Atome treiben im
 Hyperraum.":GOTO 4500
13020 IF o=47 AND (spieler=12 OR spieler
=2) AND tm$="644664" THEN spieler=2:GOTO
 1080
13021 IF o=47 AND (spieler=12 OR spieler
=2) AND tm$="644663" THEN spieler=12:GOT
O 1080
13030 IF o=22 AND spieler=1 THEN fl(1)=-
1:PRINT ok$:GOTO 1080
13900 PRINT m$(5):GOTO 1080
14000 IF o=13 AND spieler=7 AND ob(13)=7
 THEN PRINT ok$:ob(12)=0:ob(36)=-1:fl(2)
=-1:GOTO 1080
20000 WINDOW SWAP 0,1:CLS:PRINT"Transmis
sion received:":PRINT
20010 PRINT"Terra an Raumschiff CC 464:"
:PRINT
```

```
20020 PRINT"Atomarer Unfall auf PIA III"
20030 PRINT"Medikament Cibarad 92 wird":
PRINT"dringend benoetigt.":PRINT"Versorg
ungsstation 89011 anfliegen.":PRINT:PRIN
T"Koordinaten:"
20040 PRINT"89011  : 64-46-64":PRINT"
      Password: 220559":PRINT"PIA III: 1
5-10-63":PRINT"STATUS: **Fehlfunktion**"
20050 eingabe$=INKEY$:IF eingabe$="" THE
N 20050 ELSE WINDOW SWAP 1,0:alt=0:GOTO
1080
20200 WINDOW SWAP 0,1:CLS:PRINT"Status C
heck:":PRINT
20210 IF fl(1)=0 THEN PRINT"Autopilot OF
F" ELSE PRINT"Autopilot ON"
20220 IF fl(2)=0 THEN PRINT"CPU defekt"
ELSE PRINT"All Systems Go"
20230 IF fl(1)=0 THEN PRINT:PRINT"ATTENT
ION !":PRINT"Kollisionswarnung !"
20240 IF fl(2)=-1 THEN PRINT"Standort: 6
44663"
20250 eingabe$=INKEY$:IF eingabe$="" THE
N 20250 ELSE alt=0:WINDOW SWAP 1,0:GOTO
1080
21300 PLOT 320, 180:DRAW 320, 230:DRAW 4
00, 230:DRAW 400, 180:DRAW 240, 180:DRAW
 240, 230:DRAW 320, 230:PLOT 240, 230:DR
AW 260, 250:PLOT 260, 250:DRAW 380, 250:
DRAW 400, 230:PLOT 370, 250:DRAW 370, 26
0:DRAW 270, 260
21301 DRAW 270, 250:PLOT 250, 260:DRAW 2
50, 320:DRAW 390, 320:DRAW 390, 260:DRAW
 250, 260:PLOT 0, 190:DRAW 240, 190:PLOT
 400, 190:DRAW 480, 190:PLOT 480, 400:DR
AW 480, 190:DRAW 550, 120:PLOT 520, 400:
DRAW 550, 370
21302 DRAW 550, 120:DRAW 560, 120:PLOT 5
50, 370:DRAW 640, 280:PLOT 560, 360:DRAW
 560, 120:PLOT 330, 220:DRAW 390, 220:DR
AW 390, 190:DRAW 330, 190:DRAW 330, 220:
PLOT 310, 220:DRAW 250, 220:DRAW 250, 19
```

```
0:DRAW 310, 190
21303 DRAW 310, 220:PLOT 257, 237:DRAW 3
14, 237:DRAW 314, 246:DRAW 266, 246:DRAW
 257, 237:PLOT 326, 237:DRAW 383, 237:DR
AW 374, 246:DRAW 326, 246:DRAW 326, 237:
PLOT 332, 243:PLOT 338, 243:PLOT 347, 24
3:DRAW 347, 240
21304 PLOT 356, 240:DRAW 356, 243:PLOT 3
65, 243:DRAW 365, 240:PLOT 309, 243:PLOT
 309, 240:PLOT 299, 240:PLOT 292, 240:PL
OT 287, 240:PLOT 277, 240:DRAW 277, 244
21320 RETURN
21700 PLOT 0, 380:DRAW 70, 330:DRAW 70,
190:PLOT 10, 144:DRAW 70, 190:PLOT 50, 1
44:DRAW 90, 184:DRAW 90, 404:PLOT 90, 18
4:DRAW 190, 184:PLOT 190, 154:DRAW 190,
224:DRAW 450, 224:DRAW 450, 154:DRAW 190
, 154:PLOT 190, 224
21701 DRAW 200, 244:DRAW 420, 244:DRAW 4
50, 224:PLOT 440, 164:PLOT 440, 214:PLOT
 210, 214:PLOT 210, 164:PLOT 214, 240:DR
AW 414, 240:PLOT 420, 236:DRAW 212, 236:
PLOT 210, 232:DRAW 426, 232:PLOT 398, 24
4:DRAW 398, 312
21702 DRAW 402, 316:DRAW 642, 316:PLOT 6
42, 320:DRAW 398, 320:DRAW 390, 312:DRAW
 390, 244:PLOT 450, 180:DRAW 642, 180:PL
OT 192, 232:DRAW 160, 232:DRAW 156, 228:
DRAW 156, 180:PLOT 44, 312:DRAW 20, 312:
PLOT 20, 284
21703 DRAW 44, 284:PLOT 20, 256:DRAW 44,
 256:PLOT 44, 340:DRAW 20, 340:PLOT 20,
228:DRAW 44, 228:PLOT 44, 200:DRAW 20, 2
00:PLOT 20, 172:DRAW 44, 172
21720 RETURN
21800 PLOT 0, 399:DRAW 639, 399:PLOT 639
, 359:DRAW 359, 359:DRAW 359, 159:DRAW 6
39, 159:PLOT 639, 319:DRAW 359, 319:PLOT
 399, 279:PLOT 439, 239:PLOT 519, 239:PL
OT 519, 279:PLOT 479, 279:PLOT 399, 199:
PLOT 404, 204
```

```
21801 PLOT 374, 234:PLOT 429, 289:PLOT 4
74, 244:PLOT 469, 199:PLOT 554, 189:PLOT
 589, 224:PLOT 619, 269:PLOT 589, 299:PL
OT 569, 314:PLOT 569, 269:PLOT 289, 359:
DRAW 79, 359:DRAW 79, 279:DRAW 289, 279:
DRAW 289, 359:RETURN
22000 PLOT-5, 188:DRAW 465, 188:DRAW 495
, 158:DRAW 495, 338:DRAW 575, 258:DRAW 5
75, 148:PLOT 495, 158:DRAW 505, 158:DRAW
 505, 328:PLOT 465, 188:DRAW 465, 398:PL
OT 420, 310:DRAW 400, 310:DRAW 400, 330:
DRAW 380, 330
22001 DRAW 380, 310:DRAW 360, 310:DRAW 3
60, 290:DRAW 380, 290:DRAW 380, 270:DRAW
 400, 270:DRAW 400, 290:DRAW 420, 290:DR
AW 420, 310:PLOT 320, 240:DRAW 350, 210:
DRAW 150, 210:DRAW 120, 240:DRAW 320, 24
0:PLOT 0, 250
22002 DRAW 50, 250:DRAW 50, 340:DRAW-10,
 340:PLOT 50, 340:DRAW 30, 360:DRAW 0, 3
60:PLOT 40, 300:DRAW 40, 290:RETURN
22100 PLOT 75, 189:DRAW 75, 409:PLOT 75,
 189:PLOT 33, 147:DRAW 74, 188:PLOT 234,
 188:DRAW 414, 188:DRAW 389, 248:PLOT 23
4, 188:DRAW 259, 248:DRAW 389, 248:PLOT
75, 238:DRAW 255, 238:PLOT 395, 238:DRAW
 525, 238
22101 DRAW 525, 398:PLOT 525, 238:DRAW 5
85, 178:DRAW 585, 398:PLOT 585, 178:DRAW
 595, 178:DRAW 595, 398:PLOT 385, 188:DR
AW 385, 148:DRAW 375, 148:DRAW 375, 188:
PLOT 265, 188:DRAW 265, 148:DRAW 275, 14
8:DRAW 275, 188
22105 WINDOW SWAP 0,1:LOCATE 10,4:PRINT"
 Terra ist weit -";:LOCATE 10,5:PRINT"Ma
c Donald's auch !":IF ob(40)=11 THEN LOC
ATE 19,12:PRINT CHR$(131);
22106 WINDOW SWAP 1,0
22110 RETURN
22200 PLOT 50, 398:DRAW 50, 228:DRAW 0,
178:PLOT 120, 398:DRAW 120, 298:DRAW 220
```

```
, 398:PLOT 120, 298:DRAW 110, 298:DRAW 1
10, 398:PLOT 420, 248:DRAW 420, 278:DRAW
 460, 278:DRAW 460, 248:DRAW 420, 248:PL
OT 420, 278
22201 DRAW 430, 288:DRAW 450, 288:DRAW 4
60, 278:PLOT 410, 240:DRAW 470, 240:DRAW
 480, 220:DRAW 400, 220:DRAW 410, 240:PL
OT 414, 235:PLOT 418, 235:PLOT 422, 235:
PLOT 426, 235:PLOT 430, 235:PLOT 434, 23
5:PLOT 438, 235
22202 PLOT 442, 235:PLOT 446, 235:PLOT 4
50, 235:PLOT 458, 235:PLOT 462, 235:PLOT
 466, 235:PLOT 468, 231:PLOT 465, 231:PL
OT 460, 230:PLOT 462, 226:PLOT 466, 226:
PLOT 470, 226:PLOT 453, 229:DRAW 412, 22
9:PLOT 430, 248
22203 DRAW 430, 240:PLOT 452, 240:DRAW 4
52, 248:PLOT 402, 218:DRAW 422, 198:DRAW
 422, 168:DRAW 462, 168:DRAW 462, 198:DR
AW 482, 218:PLOT 462, 198:DRAW 422, 198:
PLOT 362, 398:DRAW 362, 318:DRAW 382, 29
8:DRAW 472, 298
22204 DRAW 492, 318:DRAW 492, 398:PLOT 4
22, 338:DRAW 382, 298:DRAW 362, 318:DRAW
 422, 338:PLOT 472, 298:DRAW 432, 338:DR
AW 492, 318:PLOT 432, 338:DRAW 422, 338:
PLOT 422, 398:DRAW 422, 338:DRAW 432, 33
8:DRAW 432, 398
22205 PLOT 332, 284:DRAW 426, 284:PLOT 4
54, 284:DRAW 534, 284:DRAW 554, 304:DRAW
 524, 364:DRAW 494, 394:PLOT 332, 284:DR
AW 312, 304:DRAW 332, 364:DRAW 362, 394:
PLOT 332, 284:DRAW 332, 264:DRAW 421, 26
4:PLOT 461, 264
22206 DRAW 535, 264:DRAW 535, 285:PLOT 5
35, 264:DRAW 555, 284:DRAW 555, 304:PLOT
 333, 264:DRAW 313, 284:DRAW 313, 304:PL
OT 593, 404:DRAW 643, 354
22220 RETURN
22300 PLOT 0, 390:DRAW 130, 320:DRAW 530
, 320:DRAW 640, 390:PLOT 530, 320:DRAW 5
```

```
30, 200:PLOT 620, 145:DRAW 531, 199:DRAW
 131, 199:DRAW 131, 319:PLOT 131, 199:DR
AW 91, 179:DRAW 91, 299:DRAW 1, 339:PLOT
-9, 179
22301 IF ob(50)=spieler THEN WINDOW SWAP
 0,1:LOCATE 13,13:PRINT STRING$(8,232):W
INDOW SWAP 1,0
22320 RETURN
22400 PLOT 120, 400:DRAW 120, 240:DRAW 1
90, 310:PLOT 190, 400:DRAW 190, 270:DRAW
 320, 270:DRAW 320, 350:DRAW 300, 370:DR
AW 250, 370:DRAW 220, 400:PLOT 250, 370:
DRAW 290, 410:PLOT 300, 370:DRAW 330, 40
0:PLOT 320, 350
22401 DRAW 370, 400:PLOT 320, 270:DRAW 4
50, 400:PLOT 420, 400:DRAW 330, 310:DRAW
 330, 290:DRAW 440, 400:PLOT 310, 280:DR
AW 200, 280:DRAW 200, 300:DRAW 310, 300:
DRAW 310, 280:PLOT 590, 400:DRAW 620, 37
0:DRAW 620, 400
22402 DRAW 623, 370:DRAW 623, 400:PLOT 6
23, 370:DRAW 638, 355
22420 RETURN
22600 PLOT 160, 144:DRAW 220, 184:DRAW 4
60, 184:DRAW 520, 144:PLOT 460, 184:DRAW
 460, 384:DRAW 500, 404:PLOT 460, 384:DR
AW 220, 384:DRAW 160, 404:PLOT 220, 384:
DRAW 220, 184:PLOT 280, 189:DRAW 410, 18
9:DRAW 415, 194
22601 DRAW 415, 369:DRAW 410, 374:DRAW 2
75, 374:DRAW 270, 369:PLOT 270, 369:DRAW
 270, 194:DRAW 275, 189:DRAW 280, 189:PL
OT 425, 284:DRAW 425, 274:DRAW 430, 274:
DRAW 430, 284:DRAW 425, 284:PLOT 428, 28
0:PLOT 428, 279
22610 IF ((spieler=16 AND durchgang(16,4
)<>1) OR spieler=9 OR spieler=6) THEN PL
OT 343,188:DRAW 343,375:GOTO 22613
22611 PLOT 274, 241:DRAW 329, 241:DRAW 3
29, 266:DRAW 272, 266:PLOT 328, 266:DRAW
 322, 272:DRAW 270, 272:DRAW 270, 276
```

```
22612 DRAW 318, 276:DRAW 318, 272:PLOT 3
18, 276:DRAW 322, 276:DRAW 322, 304:DRAW
 270, 304:PLOT 328, 250:DRAW 350, 250:DR
AW 350, 374:PLOT 350, 250:DRAW 396, 226:
DRAW 396, 374
22613 RETURN
22700 PLOT 60, 398:DRAW 60, 198:DRAW 10,
 148:PLOT 60, 248:DRAW 260, 248:PLOT 260
, 398:DRAW 260, 248:DRAW 360, 348:PLOT 3
60, 398:DRAW 360, 248:DRAW 480, 248:PLOT
 480, 398:DRAW 480, 188:DRAW 640, 188
22720 RETURN
23000 PLOT 0, 190:DRAW 430, 190:PLOT 430
, 400:DRAW 430, 190:PLOT 430, 191:DRAW 4
75, 146:PLOT 493, 146:DRAW 493, 362:DRAW
 583, 272:DRAW 583, 144:PLOT 498, 355:DR
AW 498, 145
23020 RETURN
23100 PLOT 120, 400:DRAW 120, 230:DRAW 3
3, 143:PLOT-3, 170:DRAW 66, 239:DRAW 60,
 239:DRAW-3, 176:PLOT 60, 239:DRAW 60, 3
98:PLOT 66, 398:DRAW 66, 239:PLOT 32, 27
3:DRAW 2, 273:PLOT 2, 298:DRAW 32, 298:P
LOT 32, 323
23101 DRAW 2, 323:PLOT 2, 348:DRAW 32, 3
48:PLOT 32, 373:DRAW 2, 373:PLOT 2, 398:
DRAW 32, 398:PLOT 32, 248:DRAW 2, 248:PL
OT 2, 223:DRAW 32, 223:PLOT 2, 198:DRAW
17, 198:PLOT 267, 208:DRAW 267, 178:DRAW
 497, 208
23102 DRAW 497, 178:DRAW 267, 208:PLOT 2
57, 208:DRAW 507, 208:DRAW 517, 218:DRAW
 517, 308:DRAW 507, 318:DRAW 257, 318:DR
AW 247, 308:DRAW 247, 218:DRAW 257, 208:
PLOT 267, 218:DRAW 497, 218:DRAW 497, 30
8:DRAW 267, 308
23103 DRAW 267, 218:PLOT 247, 248:DRAW 2
27, 248:PLOT 227, 246:DRAW 247, 246:PLOT
 234, 244:DRAW 234, 238:DRAW 234, 254:PL
OT 230, 254:DRAW 238, 254:PLOT 121, 230:
DRAW 247, 230:PLOT 516, 230:DRAW 636, 23
```

```
0:PLOT 519, 219
23104 DRAW 519, 225:PLOT 426, 318:DRAW 4
26, 366:DRAW 477, 366:DRAW 480, 369:DRAW
 426, 369:DRAW 423, 366:DRAW 423, 321:PL
OT 396, 321:DRAW 396, 327:DRAW 393, 327:
DRAW 393, 321
23120 RETURN
23200 GOSUB 31200:GOSUB 31400:GOSUB 3130
0
23300 GOSUB 31200:IF fl(5) AND spieler=2
3 THEN GOSUB 31400
23305 IF fl(6) AND spieler=25 THEN GOSUB
 31400
23310 RETURN
23400 GOSUB 31200:GOSUB 31300:GOSUB 3140
0:RETURN
23600 GOSUB 31200:GOSUB 31400:RETURN
23700 GOSUB 31200:GOSUB 31400:RETURN
24000 PLOT 0, 160:DRAW 60, 160:DRAW 60,
190:DRAW 0, 190:PLOT 60, 160:DRAW 120, 2
20:PLOT 120, 220:DRAW 120, 250
24001 DRAW 60,.190:PLOT 120, 250:DRAW 0,
 250:PLOT 120, 230:DRAW 460, 230:PLOT 54
6, 144:DRAW 460, 230:DRAW 460, 400:PLOT
490, 370:DRAW 490, 220:DRAW 500, 220:DRA
W 500, 360:PLOT 490, 370:DRAW 570, 290:D
RAW 570, 150
24002 PLOT 490, 220:PLOT 566, 144:DRAW 4
90, 220:PLOT 500, 220:DRAW 571, 150:DRAW
 501, 220:PLOT 421, 300:DRAW 411, 300:DR
AW 412, 299:DRAW 419, 299:DRAW 419, 302:
DRAW 413, 302:DRAW 414, 304:DRAW 416, 30
4:DRAW 416, 297
24003 DRAW 415, 297:RETURN
30000 PAPER 0:CLS#1:IF spieler>20 THEN 3
0030
30005 IF spieler>10 THEN 30020
30010 ON spieler GOSUB 21300,22200,21300
,21400,21500,22600,21700,21800,22600,220
00
30011 PAPER 2:RETURN
```

```
30020 ON spieler-10 GOSUB 22100,22200,22
300,22400,22300,22600,22700,22700,22700,
23000
30021 PAPER 2:RETURN
30030 ON spieler-20 GOSUB 23100,23200,23
300,23400,23300,23600,23700,22600,23900,
24000
30031 PAPER 2:RETURN
31200 PLOT 180, 400:PLOT 180, 144:DRAW 1
80, 414:PLOT 450, 144:DRAW 450, 414:PLOT
 210, 354:DRAW 270, 354:PLOT 360, 354:DR
AW 420, 354:PLOT 420, 294:DRAW 360, 294:
PLOT 270, 294:DRAW 210, 294:PLOT 210, 23
4
31201 DRAW 270, 234:PLOT 360, 234:DRAW 4
20, 234:PLOT 420, 174:DRAW 360, 174:PLOT
 270, 174:DRAW 210, 174:RETURN
31300 PLOT 360, 234:DRAW 420, 234:PLOT 4
20, 174:DRAW 360, 174:PLOT 270, 174:DRAW
 210, 174:PLOT 140, 344:DRAW 140, 184:DR
AW 90, 144:DRAW 90, 384:DRAW 140, 344:DR
AW 130, 344:DRAW 130, 184:DRAW 140, 184
31301 DRAW 130, 184:DRAW 90, 154:PLOT 13
0, 344:DRAW 90, 374:RETURN
31400 PLOT 477, 344:PLOT 477, 344:DRAW 4
77, 179:DRAW 527, 139:DRAW 527, 384:DRAW
 477, 344:PLOT 477, 344:PLOT 487, 344:DR
AW 477, 344:PLOT 527, 379:DRAW 487, 344:
DRAW 487, 179:DRAW 527, 144:PLOT 487, 17
9:DRAW 477, 179:RETURN
51500 WINDOW SWAP 0,1:CLS:PRINT CHR$(150
);STRING$(22,154);CHR$(156)
51510 PRINT CHR$(149);"  ID - Karte# 435
GER  ";CHR$(149)
51520 PRINT CHR$(149);"        fuer
      ";CHR$(149)
51530 PRINT CHR$(149);"      High Res
      ";CHR$(149)
51540 PRINT CHR$(149);STRING$(22,32);CHR
$(149)
51550 PRINT CHR$(149);"   Security Level
```

```
 I    ";CHR$(149)
51560 PRINT CHR$(149);"    (c) TERRA  IX.
84    ";CHR$(149)
51570 PRINT CHR$(147);STRING$(22,154);CH
R$(153)
51580 eingabe$=INKEY$:IF eingabe$="" THE
N 51580 ELSE WINDOW SWAP 1,0:alt=0:GOTO
1080
```

ADVENTURE EDITOR

ADVENTURE INTERPRETER

```
5 REM Adventure Editor
6 REM (c) 1984 by J. Walkowiak
7 REM___________________________________
10 MODE 1:CLEAR:DEFINT a-z:DIM ra$(60),o
b$(60),rn$(60),ob(60),ve$(30),m$(60),ac$
(30,60),bc$(30,60),ad$(20),du(60,6)
20 ar=0:ao=0:av=0:am=0:af=0:i2=1:l3=1:m1
$="         ADVENTURE EDITOR   Ver 1.0":m2
$=STRING$(2,10):m2$=m2$+"Was wuenschen S
ie":m3$=STRING$(40,154):m4$=STRING$(40,3
2)
30 CLS:PRINT m1$:PRINT m3$;
40 PRINT:PRINT:INPUT"Welches Adventure w
ollen Sie bearbeiten ";na$:PRINT:INPUT"V
ersion Nr. ";ve$:na$=UPPER$(na$)
50 PRINT:INPUT"Copyright - Vermerk ";cr$
:PRINT:PRINT m3$
60 GOSUB 6000
200 CLS:PRINT m1$:PRINT m3$
210 PRINT TAB(10)"0 - Daten"
220 PRINT TAB(10)"1 - Raeume eingeben"
230 PRINT TAB(10)"2 - Objekte eingeben"
240 PRINT TAB(10)"3 - Verben eingeben"
250 PRINT TAB(10)"4 - Startpositionen"
260 PRINT TAB(10)"5 - Raeume verbinden"
270 PRINT TAB(10)"6 - Mitteilungen einge
ben"
280 PRINT TAB(10)"7 - Bedingungen & Akti
onen"
290 PRINT TAB(10)"8 - Datentraeger"
300 PRINT TAB(10)"9 - Programmende"
310 PRINT m3$
320 PRINT TAB(10)"Bitte waehlen Sie"
330 GOSUB 10000
340 IF a=9 THEN en=-1:GOTO 5100
350 ON a+1 GOTO 3000,1100,1200,1300,1400
,1500,4000,1600,5000,6000
499 REM ****** UNTERPROGRAMM EIN/AUSGABE
500 CLS
510 LOCATE 1,1:PRINT m4$:LOCATE 1,1:PRIN
T t1$
```

```
520 z=z+1
530 LOCATE 25,1:PRINT t2$;z
540 PRINT m3$
590 LOCATE 1,25:PRINT t3$;:eingabe$="":L
INE INPUT eingabe$
600 IF LEN(eingabe$)=0 THEN GOTO 200
610 IF a=2 THEN PRINT:LOCATE 1,25:INPUT"
Rufname ";rn$(z):rn$(z)=UPPER$(rn$(z)):P
RINT
620 PRINT
640 RETURN
1100 CLS:t1$="Orte eingeben ":t2$=" Raum
 Nr.:":t3$="Ich bin "
1105 z=ar
1110 GOSUB 510:       REM raeume drucken
1120 ra$(z)=eingabe$
1125 ar=z
1130 GOTO 1110
1199 REM ************* EINGABE ORTE ENDE
1200 CLS:t1$="Objekte eingeben ":t2$=" O
bjekt Nr.:":t3$="Ich sehe "
1205 z=ao
1210 GOSUB 510
1220 ob$(z)=eingabe$
1225 ao=z
1230 GOTO 1210
1290 REM ********** ENDE EINGABE OBJEKTE
1300 CLS:t1$="Verben eingeben":t2$="Verb
 Nr.:":t3$=""
1305 z=av
1310 GOSUB 510
1320 ve$(z)=UPPER$(eingabe$)
1330 av=z
1360 GOTO 1310
1390 REM ********** ENDE EINGABE VERBEN
1400 CLS
1410 lp=13
1420 FOR i=lp TO ao
1430 GOSUB 12000:REM liste drucken
1440 LOCATE 1,23:PRINT"Objekt ";rn$(i);"
 in Raum Nr.";:INPUT ob(i)
```

```
1450 lp=i+1
1460 CLS
1470 NEXT i
1480 GOTO 200
1500 FOR r1=i2 TO ar
1510 CLS
1515 PRINT
1520 GOSUB 12000
1530 PRINT"Raum";r1;"fuehrt im Norden na
ch";:INPUT du(r1,1)
1540 PRINT"Raum";r1;"fuehrt im Sueden na
ch";:INPUT du(r1,2)
1550 PRINT"Raum";r1;"fuehrt im Westen na
ch";:INPUT du(r1,3)
1560 PRINT"Raum";r1;"fuehrt im Osten  na
ch";:INPUT du(r1,4)
1570 PRINT"Raum";r1;"fuehrt oben  nach";
:INPUT du(r1,5)
1580 PRINT"Raum";r1;"fuehrt unten nach";
:INPUT du(r1,6)
1590 NEXT r1:GOTO 200
1600 CLS
1605 nb=0
1607 nb=nb+1
1608 IF nb>av THEN GOTO 200
1609 rn$(0)="<weiter>"
1610 CLS:FOR i=0 TO ao
1620 PRINT i;rn$(i);"\";
1630 IF POS(#0)+LEN(rn$(i+1))>38 THEN PR
INT
1640 NEXT i
1650 PRINT:PRINT m3$:PRINT CHR$(10)
1700 PRINT"Aktion auf Verb ";ve$(nb);" u
nd Objekt Nr.";
1710 INPUT ob
1715 IF ob=0 THEN GOTO 1607
1800 CLS
1810 PRINT"Bitte Bedingungen fuer die Ak
tion ":PRINT ve$(nb);" ";rn$(ob);" einge
ben"
1815 PRINT m3$
```

```
1820 PRINT:PRINT:PRINT TAB(5)" R  - Obje
kt ist im Raum":PRINT
1830 PRINT TAB(5)" I  - Objekt ist im In
vebtory":PRINT
1840 PRINT TAB(5)" N  - Objekt ist nicht
 im Raum":PRINT
1850 PRINT TAB(5)" Fx - Flag ist gesetzt
1860 PRINT:PRINT TAB(5)" Gx - Flag ist g
eloescht":PRINT
1870 PRINT TAB(5)" Sxx- Spieler ist im R
aum xx"
1880 LOCATE 1,20:PRINT m3$;:PRINT"Alter
Code ===> ";bc$(nb,ob):PRINT
1900 INPUT"Bedingung ";bc$(nb,ob):bc$(nb
,ob)=UPPER$(bc$(nb,ob))
1905 CLS:PRINT ve$(nb);" ";ob$(ob);" bew
irkt, wenn:"
1906 PRINT bc$(nb,ob);" erfuellt, folgen
de Aktion:":PRINT m3$;
1910 PRINT TAB(5)"V  - ";rn$(ob);" versc
hwindet":PRINT
1920 PRINT TAB(5)"I  - ";rn$(ob);" kommt
 ins INV":PRINT
1930 PRINT TAB(5)"Nxx- Objekt xx erschei
nt neu":PRINT
1940 PRINT TAB(5)"Dxy- Durchgang nach Ra
um x":PRINT
1950 PRINT TAB(5)"Sxx- Spieler nach Raum
 xx":PRINT
1960 PRINT TAB(5)"Fx - Flag x setzen":PR
INT
1970 PRINT TAB(5)"Lx - Flag x loeschen":
PRINT
1980 PRINT TAB(5)"Mxx- Meldung xx ausgeb
en":PRINT
1990 PRINT TAB(5)"T  - Spieler stirbt":P
RINT
2000 PRINT TAB(5)"E  - Ende, da gewonnen
"
2005 PRINT m3$;
2010 PRINT"Alter Code ===> ";ac$(nb,ob)
```

```
2020 INPUT"Aktion ";ac$(nb,ob):ac$(nb,ob
)=UPPER$(ac$(nb,ob))
2030 GOTO 1610
2040 REM ****** ENDE AKTIONEN & BEDINGUN
G
2080 GOSUB 10000
2999 REM ********************* KENNDATEN
3000 CLS:PRINT"Kenndaten des Adventures
";na$
3010 PRINT m3$:PRINT
3020 PRINT" Raeume: ";ar
3030 PRINT" Objekte:";ao
3040 PRINT" Verben: ";av
3050 PRINT" Beding: ";nb
3060 PRINT" Mittlg.:";am
3070 IF lo THEN PRINT" ";af;"benutzte Fl
ags"
3080 PRINT m3$
3090 PRINT"Spielbeginn in Raum:
";sp
3100 PRINT:PRINT"erforderliche Eingabela
enge: ";wl
3110 GOSUB 10000
3120 GOTO 200
3999 REM ******************* Mitteilungen
4000 CLS:PRINT"Mitteilungen eingeben"
4010 PRINT m3$
4020 am=am+1
4030 PRINT am;:INPUT m$(am)
4040 IF LEN(m$(am))=0 THEN am=am-1:GOTO
200
4050 GOTO 4020
4060 REM ************* Ende MITTEILUNGEN
4999 REM DATENTRAEGER: speichern/laden
5000 CLS:PRINT"1 - Sicherheitskopie"
5010 PRINT:PRINT"2 - Datei zur Bearbeitu
ng laden"
5020 GOSUB 10000
5030 IF a<1 OR a>2 THEN GOTO 5020
5040 ON a GOTO 5100,5500
5100 CLS:PRINT"Bitte warten.":PRINT:PRIN
```

```
T"Speicherung laeuft ..."
5110 OPENOUT na$
5120 PRINT#9,na$:PRINT#9,ve$:PRINT#9,cr$
:PRINT#9,wl
5130 PRINT#9,ar:PRINT#9,ao:PRINT#9,av:PR
INT#9,am:PRINT#9,sp:PRINT#9,af
5140 FOR i=1 TO ar:PRINT#9,ra$(i):NEXT i
5150 FOR i=1 TO ao:PRINT#9,ob$(i):PRINT#
9,rn$(i):PRINT#9,ob(i):NEXT i
5160 FOR i=1 TO av:PRINT#9,ve$(i):NEXT i
5170 FOR i=1 TO am:PRINT#9,m$(i):NEXT i
5180 FOR x=1 TO ar
5190 FOR y=1 TO 6
5200 PRINT#9,du(x,y)
5210 NEXT y
5220 NEXT x
5230 FOR x=1 TO av
5240 FOR y=1 TO ao
5250 PRINT#9,bc$(x,y)
5260 NEXT y
5270 NEXT x
5280 FOR x=1 TO av
5290 FOR y=1 TO ao
5300 PRINT#9,ac$(x,y)
5310 NEXT y
5320 NEXT x
5330 CLOSEOUT:IF en THEN CLS:END
5340 GOTO 200
5500 CLS:PRINT"Bitte warten.":PRINT:PRIN
T"Daten werden geladen ..."
5510 OPENIN na$
5520 INPUT#9,na$:INPUT#9,ve$:INPUT#9,cr$
:INPUT#9,wl
5530 INPUT#9,ar:INPUT#9,ao:INPUT#9,av:IN
PUT#9,am:INPUT#9,sp:INPUT#9,af
5540 FOR i=1 TO ar:INPUT#9,ra$(i):NEXT i
5550 FOR i=1 TO ao:INPUT#9,ob$(i):INPUT#
9,rn$(i):INPUT#9,ob(i):NEXT i
5560 FOR i=1 TO av:INPUT#9,ve$(i):NEXT i
5570 FOR i=1 TO am:INPUT#9,m$(i):NEXT i
5580 FOR x=1 TO ar
```

```
5590 FOR y=1 TO 6
5600 INPUT#9,du(x,y)
5610 NEXT y
5620 NEXT x
5630 FOR x=1 TO av
5640 FOR y=1 TO ao
5650 INPUT#9,bc$(x,y)
5660 NEXT y
5670 NEXT x
5680 FOR x=1 TO av
5690 FOR y=1 TO ao
5700 INPUT#9,ac$(x,y)
5710 NEXT y
5720 NEXT x
5730 CLOSEIN
5740 GOTO 200
5999 REM ***************** uebrige Daten
6000 CLS:PRINT"Bitte geben Sie die noch
notwendigen    Kontrolldaten ein:":PRINT
 m3$
6010 PRINT:PRINT:INPUT"Wortlaenge";wl
6020 PRINT:INPUT"Startraum ";sp
6030 PRINT:INPUT"Anzahl Flags";af
6040 RETURN
10000 a$=INKEY$:IF a$="" THEN 10000
10010 a=VAL(a$):RETURN
12000 LOCATE 1,1
12010 PRINT"Liste der moeglichen Raeume:
":PRINT m3$
12020 FOR i1=1 TO ar
12030 PRINT i1;ra$(i1);
12040 IF POS(#0)+LEN(ra$(i1+1)) >38 THEN
 PRINT
12050 NEXT i1
12060 PRINT
12070 RETURN
```

```
1 REM ********************************
2 REM * Adventure System 1.0         *
3 REM * Adventureinterpreter         *
4 REM * (c) 1983,1984 by J. Walkowiak *
5 REM ********************************
6 CLS:PRINT"          Adventure System
1.0":PRINT:PRINT"            by Joerg Wal
kowiak":PRINT:PRINT:PRINT:PRINT"
 (c) 1984 by DATA BECKER":DEFINT a-z
10 FOR i=1 TO 5000:NEXT
40 CLEAR:CLS:INPUT"Welches Adventure wol
len Sie spielen    ";na$
50 start=-1:GOSUB 900
60 INPUT#9,na$:CLS:PRINT na$;:INPUT#9,ve
$:PRINT"Version ";ve$:INPUT#9,cr$:INPUT#
9,wl:PRINT cr$
70 INPUT#9,ar:INPUT#9,ao:INPUT#9,av:INPU
T#9,am:INPUT#9,spieler:INPUT#9,af
75 DIM raum$(ar),ob$(ao),rn$(ao),ob(ao),
verb$(av),m$(am),fl(af),ac$(av,ao),bc$(a
v,ao),ad$(20),durchgang(ar,6)
80 FOR i=1 TO ar:INPUT#9,raum$(i):NEXT i
90 FOR i=1 TO ao:INPUT#9,ob$(i):INPUT#9,
rufname$(i):INPUT#9,ob(i):rufname$(i)=LE
FT$(rufname$(i),wl):NEXT i
100 FOR i=1 TO av:INPUT#9,verb$(i):verb$
(i)=LEFT$(verb$(i),wl):NEXT i
110 FOR i=1 TO am:INPUT#9,m$(i):NEXT i
120 FOR x=1 TO ar:FOR y=1 TO 6
130 INPUT#9,durchgang(x,y)
140 NEXT y:NEXT x
150 FOR x=1 TO av:FOR y=1 TO ao
160 INPUT#9,bc$(x,y)
170 NEXT y:NEXT x
180 FOR x=1 TO av:FOR y=1 TO ao
190 INPUT#9,ac$(x,y)
200 NEXT y:NEXT x
210 CLOSEIN
880 GOTO 1000
900 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
1:INK 1,25
```

```
910 PRINT"       CPC - Ventures"
920 PRINT TAB(15)CHR$(164);" 1984  by J"
;CHR$(178);"rg Walkowiak"
925 PRINT STRING$(40,208):PRINT"Stellen
Sie  sich einen Roboter vor, denSie  mit
 zahlreichen  Kommandos  steuernkoennen.
"
930 PRINT"Ich bin dieser  Roboter,  und
ich  werdemich fuer Sie den Gefahren der
 verwegen-sten Abenteuer aussetzen."
940 PRINT"Damit Sie  mich sinnvoll  agie
ren lassenkoennen,  werde ich  Ihnen die
 Situationin der ich  mich gerade befind
e, jeweilsgenau beschreiben."
950 PRINT"Anschliessend  sagen  Sie  mir
  mit zweiWorten,  wie  beispielsweise
UNTERSUCHETUER, NIMM MESSER, was ich tun
 soll.":PRINT
960 PRINT"Darueber hinaus verstehe ich d
ie Befehle     INVENTUR, INSTRUKTIONEN u
nd ENDE.":IF NOT start THEN GOTO 980
970 start=0:PRINT STRING$(40,210) :PRINT
" Bitte warten ...  Spiel wird geladen !
":pg$="!"+na$:OPENIN pg$
980 INK 3,12,24:INK 2,24,12:PEN 3: PRINT
"           <TASTE>";:PEN 2:PRINT" drueck
en";:PEN 1:PRINT" ..."
990 eingabe$=INKEY$:IF eingabe$="" THEN
990 ELSE CLS:MODE 1:INK 1,2:INK 2,14:INK
 3,26:RETURN
1000 MODE 1:PAPER 0:INK 0,0:BORDER 0:PEN
 1:INK 1,2:PEN 2:INK 2,14:PEN 3:INK 3,26

1010 leerzeile$=STRING$(40,32)
1020 DATA Norden, Sueden, Westen, Osten,
 Oben, Unten
1030 FOR richtung=1 TO 6
1040 READ richtung$(richtung)
1050 NEXT richtung
1070 CLS
1080 PRINT:PRINT
```

```
1090 LOCATE 1,1
1100 FOR zeile=1 TO 10
1110 PRINT leerzeile$;
1120 NEXT zeile
1130 LOCATE 1,1:PEN 1
1140 PRINT"Ich bin ";
1150 PRINT raum$(spieler);:PRINT"."
1160 PRINT"Ich sehe ";:gedruckt=0
1170 FOR i=1 TO ao
1180 IF ob(i)<>spieler THEN 1210
1190 IF POS(#0)+LEN(ob$(i))+2<=39 THEN P
RINT ob$(i);", ";:GOTO 1205
1200 IF POS(#0)+LEN(ob$(i))+2>39 THEN PR
INT:GOTO 1190
1205 gedruckt=-1
1210 NEXT i
1215 IF NOT gedruckt THEN PRINT"nichts b
esonderes  ";
1220 PRINT CHR$(8);CHR$(8);"."
1230 PRINT leerzeile$:PEN 2
1240 PRINT"Ich kann nach ";:gedruckt=0
1250 FOR RICHTUNG=1 TO 6
1260 IF durchgang(spieler,richtung)=0 TH
EN GOTO 1300 ELSE gedruckt=-1
1270 IF POS(#0)=15 THEN PRINT richtung$(
richtung);:GOTO 1300
1280 IF POS(#0)+LEN(richtung$(richtung))
<38 THEN PRINT", ";richtung$(richtung);:
GOTO 1300
1290 PRINT",":PRINT richtung$(richtung);
:GOTO 1300
1300 NEXT richtung
1310 IF gedruckt=0 THEN PRINT"nirgendwo"
;
1320 PRINT".":PEN 3
1330 PRINT STRING$(40,154);
1390 PEN 3:LOCATE 1,25:INPUT"Was soll ic
h tun";eingabe$:eingabe$=UPPER$(eingabe$
):PEN 2
1400 IF LEN(eingabe$)>2 THEN 1500
1410 IF eingabe$="N" AND durchgang(spiel
```

```
er,1)<>0 THEN spieler=durchgang(spieler,
1):PRINT"O.k.":GOTO 1080
1420 IF eingabe$="S" AND durchgang(spiel
er,2)<>0 THEN spieler=durchgang(spieler,
2):PRINT"O.k.":GOTO 1080
1430 IF eingabe$="W" AND durchgang(spiel
er,3)<>0 THEN spieler=durchgang(spieler,
3):PRINT"O.k.":GOTO 1080
1440 IF eingabe$="O" AND durchgang(spiel
er,4)<>0 THEN spieler=durchgang(spieler,
4):PRINT"O.k.":GOTO 1080
1450 IF eingabe$="OB" AND durchgang(spie
ler,5)<>0 THEN spieler=durchgang(spieler
,5):PRINT"O.k.":GOTO 1080
1460 IF eingabe$="U" AND durchgang(spiel
er,6)<>0 THEN spieler=durchgang(spieler,
6):PRINT"O.k.":GOTO 1080
1470 IF LEN(eingabe$)<3 THEN PRINT"Dahin
 fuehrt kein Weg !":GOTO 1080
1480 IF LEN(eingabe$)>8 THEN GOTO 2000
1499 REM **************** START INVENTUR
1500 IF LEFT$(eingabe$,3)<>"INV" THEN GO
TO 1600
1510 PRINT"Ich trage folgendes mit mir:"
1520 FOR objekt=1 TO ao
1530 IF ob(objekt)=-1 THEN PRINT ob$(obj
ekt)
1540 NEXT objekt
1550 GOTO 1080
1560 REM ***************** ENDE INVENTUR
1599 REM ********************** SAVE GAME
1600 IF LEFT$(eingabe$,3)<>"SAV" THEN GO
TO 1700
1620 OPENOUT "spielstand"
1625 PRINT#9,spieler
1630 FOR objekt=1 TO ao
1631 PRINT#9,ob(objekt)
1632 NEXT objekt
1635 FOR raum=1 TO ar
1636 FOR richtung=1 TO 6
1637 PRINT#9,durchgang(raum,richtung)
```

```
1638 NEXT richtung
1639 NEXT raum
1645 FOR flag=1 TO af
1646 PRINT#9,fl(flag)
1647 NEXT flag
1650 CLOSEOUT
1670 GOTO 1080
1699 REM ********************** LOAD GAME
1700 IF LEFT$(eingabe$,3)<>"LOA" THEN GO
TO 1900
1720 OPENIN"spielstand"
1725 INPUT#9,spieler
1730 FOR objekt=1 TO ao
1731 INPUT#9,ob(objekt)
1732 NEXT objekt
1735 FOR raum=1 TO ar
1736 FOR richtung=1 TO 6
1737 INPUT#9,durchgang(raum,richtung)
1738 NEXT richtung
1739 NEXT raum
1745 FOR flag=1 TO af
1746 INPUT#9,fl(flag)
1747 NEXT flag
1750 CLOSEIN
1770 GOTO 1080
1900 IF LEFT$(eingabe$,3)<>"INS" THEN GO
TO 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(eingabe$,3)<>"END" THEN GO
TO 2000 ELSE CLS
1960 PRINT"Der Autor wuenscht Ihnen mehr
 Erfolg":PRINT:PRINT"beim naechsten Mal
!":PRINT:PRINT:PRINT:END
2000 laenge=LEN(eingabe$)
2010 FOR buchstabe=1 TO laenge
2020 pruef$=MID$(eingabe$,buchstabe,1)
2030 IF pruef$<>" "THEN NEXT buchstabe
2040 everb$=LEFT$(eingabe$,wl)
2050 rl=laenge-buchstabe
2060 IF rl<0 THEN 2090
```

```
2070 eobjekt$=RIGHT$(eingabe$,rl)
2080 eobjekt$=LEFT$(eobjekt$,wl)
2090 FOR vn=1 TO av
2100 IF everb$=verb$(vn) THEN 2130
2110 NEXT vn
2120 PRINT"Das Verb verstehe ich nicht !
":GOTO 1080
2130 FOR o=1 TO ao
2140 IF eobjekt$=rufname$(o) THEN 2200
2150 NEXT o
2160 PRINT"Ich verstehe das Objekt nicht
 !":GOTO 1080
2198 REM vn & o sind verb/objekt nummern
2199 REM ********** BEDINGUNGEN ERFUELLT
2200 FOR ab=1 TO LEN(bc$(vn,o))
2210 bd$(ab)=MID$(bc$(vn,o),ab,1)
2220 NEXT ab
2250 FOR aa=1 TO LEN(ac$(vn,o))
2260 ad$(aa)=MID$(ac$(vn,o),aa,1)
2270 NEXT aa
2280 REM bedingungen in bd$
         aktionen in ad$
2290 ab=ab-1:aa=aa-1
2300 x=0:er=-1:ok=-1
2310 x=x+1:ok=(ok AND er):er=0
2320 IF x=ab+1 THEN 2500
2330 IF bd$(x)<>"R" THEN 2350
2340 IF ob(o)=spieler THEN er=-1
2345 GOTO 2310
2350 IF bd$(x)<>"I" THEN 2370
2360 IF ob(o)=-1 THEN er=-1
2365 GOTO 2310
2370 IF bd$(x)<>"N" THEN 2390
2380 IF (ob(o)<>spieler AND ob(o))<>-1)
THEN er=-1
2385 GOTO 2310
2390 IF bd$(x)<>"S" THEN 2410
2400 r1$=bd$(x+1)+bd$(x+2):x=x+2:IF spie
ler=VAL(r1$) THEN er=-1
2405 GOTO 2310
2410 IF bd$(x)<>"F" THEN 2450
```

```
2420 IF fl(VAL(bd$(x+1)))=-1 THEN 2440
2430 x=x+1:GOTO 2450
2440 er=-1:x=x+1:GOTO 2310
2450 IF bd$(x)<>"G" THEN 2310
2460 IF NOT fl(VAL(bd$(x+1))) THEN 2480
2470 x=x+1:GOTO 2310
2480 er=-1:x=x+1:GOTO 2310
2490 GOTO 2310
2500 IF NOT ok THEN PRINT"Das geht im Mo
ment nicht !":GOTO 1080
3999 REM **** ALLE BEDINGUNGEN ERFUELLT
4000 PRINT"Okay !"
4999 REM *********** AKTIONEN AUSFUEHREN
5000 x=0:er=0
5040 x=x+1
5050 IF x>AA THEN 1080
5060 IF ad$(x)<>"V" THEN 5080
5070 ob(o)=0:GOTO 5040
5080 IF ad$(x)<>"I" THEN 5100
5090 ob(o)=-1:GOTO 5040
5100 IF ad$(x)<>"N" THEN 5120
5110 GOSUB 5500:ob(pa)=spieler:ac$(vn,o)
="M00":GOTO 5040
5120 IF ad$(x)<>"F" THEN 5140
5130 GOSUB 5600:fl(pa)=-1:GOTO 5040
5140 IF ad$(x)<>"L" THEN 5160
5150 GOSUB 5600:fl(pa)=0:GOTO 5040
5160 IF ad$(x)<>"M" THEN 5180
5170 GOSUB 5500:PRINT m$(pa):GOTO 5040
5180 IF ad$(x)<>"E" THEN 5200
5190 GOTO 6000
5200 IF ad$(x)<>"D" THEN 5040
5210 GOSUB 5700
5220 durchgang(spieler,p1)=p2
5230 IF p1=1 THEN p3=2 ELSE IF p1=2 THEN
 p3=1 ELSE IF p1=3 THEN p3=4 ELSE IF p1=
4 THEN p3=3 ELSE IF p1=5 THEN p3=6 ELSE
IF p1=6 THEN p3=5
5240 du(p2,p3)=spieler:GOTO 5040
5500 r1$=ad$(x+1)+ad$(x+2):x=x+2:pa=VAL(
r1$):RETURN
```

```
5600 r1$=ad$(x+1):x=x+1:pa=VAL(r1$):RETU
RN
5700 p1=VAL(ad$(x+1)):p2=VAL(ad$(x+2)):x
=x+2:RETURN
6000 CLS:PRINT"Herzlichen Glueckwunsch !
":PRINT:PRINT
6010 PRINT"Du hast es geschafft !":PRINT
6020 PRINT STRING$(17,10):END
```

VENTUREFIX

Dieses Programm wird Sie mit allen Fragen konfrontieren, die beantwortet sein müssen, um ein komplettes, funktionsfähiges Adventureprogramm zu erzeugen.

Vor dem Start von ´Venturefix´ muß die Handlung stehen, Sie müssen eine Liste aller Räume, Gegenstände, Wörter und auch Mitteilungen erstellt haben, wie Sie auch sämtliche Aktionen bereits textuell formuliert haben sollten.

Die kreative Arbeit wird Ihnen also nicht abgenommen werden, wohl aber die Routinearbeit des Programmierens.
Sie müssen nicht einmal wissen, was ein PRINT - Befehl ist, dennoch können Sie fehlerlose BASIC - Programme erzeugen.

Venturefix wird im Dialog alle erforderlichen Daten der Reihe nach von Ihnen anfordern und Sie ständig über den Stand der Entwicklung informieren, bis schließlich ein fertiges Adventure im Stile der in diesem Buch vorgestellten Programme entstanden ist.

```
10 REM ***********************************
20 REM * Adventuregenerator VENTUREFIX *
30 REM * (c) 1984   by Joerg Walkowiak *
40 REM ***********************************

50 MODE 1:DEFINT a-z
60 PRINT"Venturefix":PRINT:PRINT"written
 by Joerg Walkowiak":LOCATE 1,24:PRINT"V
enturefix":PRINT"(c) 1984  by DATA BECKE
R, Duesseldorf";
70 FOR i=1 TO 10000:NEXT
80 WINDOW#0,1,40,1,20
90 WINDOW#1,1,40,19,25
100 GOTO 170
110 IF druck THEN PRINT#8,zeilennummer;"
 ";z$
120 z1$=STR$(zeilennummer)+" "+z$:PRINT#
1,z1$
130 PRINT#9,zeilennummer;" ";z$:zeilennu
mmer=zeilennummer+zeilenabstand:z$="":RE
TURN
140 z$=z$+h$:RETURN
150 eingabe$=INKEY$:IF eingabe$="" THEN
150 ELSE RETURN
160 CLS:PLOT 0,115:DRAW 640,115:PRINT t$
:PRINT STRING$(40,154):RETURN
170 CLS:REM ******** beginn hauptprogram
m
180 INPUT"Wie soll das Adventure heissen
";eingabe$:eingabe$=UPPER$(eingabe$)
190 IF LEN(eingabe$)>8 THEN PRINT"Bitte
nicht mehr als 8 Buchstaben !":GOTO 180
200 adventure$=eingabe$+".BAS"
210 PRINT:INPUT"Soll gleichzeitig ein Li
sting auf einem angeschlossenem Drucker
erzeugt            werden ";eingabe$
220 IF (eingabe$="j" OR eingabe$="J") TH
EN druck=-1
230 CLS:INPUT"Name des Autors";autor$
240 PRINT adventure$;:INPUT", Version ";
version$
```

```
250 INPUT"Datum ";datum$
260 PRINT:PRINT:LINE INPUT"Legen Sie ein
e leere Diskette/Cassette  ein und druec
ken Sie <ENTER> ...",eingabe$
270 t$="Kontrolldaten eingeben"
280 CLS#1:GOSUB 160
290 INPUT"Wie viele Raeume       ";ar
300 INPUT"Wie viele Objekte      ";ao
310 INPUT"Wie viele Verben       ";av
320 INPUT"Schaetzwert: Flags     ";af
330 INPUT"Spielstart in Raum     ";spieler
340 PRINT:INPUT"signifikante Wortlaenge
";wl
350 t$="Teil   I : Initialisierung":GOSU
B 160
360 PRINT"- Datei oeffnen":adventure$="!
"+ UPPER$(adventure$):OPENOUT adventure$
370 PRINT"- Programmkopf":zeilennummer=1
:zeilenabstand=1:z$="REM "+LEFT$(adventu
re$,(LEN(adventure$)-4))+", Version "+ve
rsion$:GOSUB 110
380 z$="REM (c) "+datum$:GOSUB 110
390 z$="REM by  "+autor$:GOSUB 110
400 z$="REM produziert unter Zuhilfenahm
e von:":GOSUB 110
410 z$="REM VENTUREFIX, (c) 1984 by J. W
alkowiak":GOSUB 110
420 z$="REM"+STRING$(35,154):GOSUB 110
430 PRINT"- Titel":z$="REM hier Titelbil
d programmieren":zeilennummer=10
440 PRINT"- Kenndaten":zeilennummer=150:
z$="ar="+STR$(ar)+":ao="+STR$(ao)+":av="
+STR$(av)+":af="+STR$(af):zeilenabstand=
10:GOSUB 110
450 z$="spieler="+STR$(spieler):GOSUB 11
0
460 z$="wl="+STR$(wl):GOSUB 110
470 PRINT"- Felder dimensionieren"
480 z$="DIM raum$(ar),durchgang(ar,6),ob
$(ao),rufname$(ao),ob(ao),verb$(av)":zei
lennummer=190:GOSUB 110
```

```
490 z$="REM ************************* VE
RBEN":GOSUB 110:zeilenabstand=1
500 t$="Geben Sie nun bitte jeweils ein
Verb einund druecken Sie <ENTER>:":GOSUB
 160
510 FOR i=1 TO av
520 PRINT"Verb ";i;:INPUT"lautet ";verb$
530 z$="DATA "+verb$
540 GOSUB 110
550 NEXT i
560 zeilennummer=300:z$="REM ***********
******** GEGENSTAENDE":GOSUB 110
570 CLS:t$="Geben Sie nun bitte jeweils
ein Objekt  einund druecken Sie <ENTER>:
":GOSUB 160
580 FOR i=1 TO ao
590 CLS:GOSUB 160
600 LOCATE 1,5:PRINT"Gegenstand Nr.: ";i
610 INPUT"Ich sehe ";objekt$
620 INPUT"Rufname  ";rufname$
630 PRINT
640 INPUT"in Raum Nr";raum$
650 z$="DATA "+objekt$+","+rufname$+","+
raum$:GOSUB 110
660 NEXT i
670 t$="Geben Sie nun die Beschreibung e
ines     jeden Raumes, sowie die Nummern
der von ihm aus zu erreichenden Raeume e
in.":GOSUB 160
680 zeilennummer=500
690 z$="REM ************* RAUMBESCHREIBU
NGEN":GOSUB 110
700 FOR i=1 TO ar
710 CLS:GOSUB 160
720 LOCATE 1,5:PRINT"Raum Nr.: ";i
730 INPUT"Ich bin ";raum$
740 PRINT:PRINT"Dieser Raum fuehrt"
750 INPUT" im Norden in Raum Nr. ";n$
760 INPUT" im Sueden in Raum Nr. ";s$
770 INPUT" im Westen in Raum Nr. ";w$
780 INPUT" im Osten in Raum Nr.  ";o$
```

```
790 INPUT" nach oben in Raum Nr.  ";ob$
800 INPUT" nach unten in Raum Nr. ";u$
810 z$="DATA "+raum$+","+n$+","+s$+","+w
$+","+o$+","+ob$+","+u$:GOSUB 110
820 NEXT i
830 t$="Auf folgende Standardmitteilunge
n        koennen Sie zugreifen":GOSUB 160
840 PRINT"m$(1): Ich sehe nichts besonde
res."
850 PRINT"m$(2): So stark bin ich nicht
!"
860 PRINT"m$(3): Wie stellst Du Dir das
vor ?"
870 PRINT"m$(4): Ich verstehe nicht, was
 Du                 meinst !"
880 PRINT:PRINT"ok$:    Okay !"
890 zeilennummer=600:z$="REM ***********
******** MITTEILUNGEN":GOSUB 110
900 z$="m$(1)="+CHR$(34)+"Ich sehe nicht
s besonderes."+CHR$(34):GOSUB 110
910 z$="m$(2)="+CHR$(34)+"So stark bin i
ch nicht !"+CHR$(34):GOSUB 110
920 z$="m$(3)="+CHR$(34)+"Wie stellst Du
 Dir das vor ?"+CHR$(34):GOSUB 110
930 z$="m$(4)="+CHR$(34)+"Ich verstehe n
icht, was Du meinst !"+CHR$(34):GOSUB 11
0
940 z$="ok$="+CHR$(34)+"Okay !"+CHR$(34)
:GOSUB 110
950 t$="Teil II":GOSUB 160
960 zeilennummer=699:z$="REM ***********
 hier evtl. 2. Titel":GOSUB 110
970 PRINT"- Initialisierungsteil schreib
en"
980 zeilennummer=800:zeilenabstand=10
990 z$="FOR i=1 TO av":GOSUB 110
1000 z$="READ verb$(i):verb$(i)=LEFT$(ve
rb$(i),wl):verb$(i)=UPPER$(verb$(i))":GO
SUB 110
1010 z$="NEXT i":GOSUB 110
1020 z$="FOR objekt=1 TO ao":GOSUB 110
```

```
1030 z$="READ ob$(objekt),rufname$(objek
t),ob(objekt):rufname$(objekt)=LEFT$(ruf
name$(objekt),wl):rufname$(objekt)=UPPER
$(rufname$(objekt)):NEXT objekt":GOSUB 1
10
1040 z$="FOR raum=1 TO ar:READ raum$(rau
m)":GOSUB 110
1050 z$="FOR richtung=1 TO 6:READ durchg
ang(raum,richtung)":GOSUB 110
1060 z$="NEXT richtung:NEXT raum":GOSUB
110
1070 PRINT"- Spielanweisung"
1080 z$="PRINT:INPUT"+CHR$(34)+"   Wuens
chen Sie Ratschlaege fuer Ihr
     weiteres Vorgehen"+CHR$(34)+";eingab
e$:eingabe$=UPPER$(eingabe$)":GOSUB 110
1090 z$="IF LEFT$(eingabe$,1)="+CHR$(34)
+"J"+CHR$(34)+"THEN GOSUB 900:GOTO 1000
 ELSE GOTO 1000":GOSUB 110
1100 z$="MODE 1:PAPER 0:INK 0,0:BORDER 0
:PEN 1:INK 1,25":GOSUB 110
1110 Z$="PRINT"+CHR$(34)+"       CPC - Ve
ntures"+CHR$(34)+":PRINT TAB(15)CHR$(164
);"+CHR$(34)+" 1984  by J"+CHR$(34)+"CHR
$(178);"+CHR$(34)+"rg Walkowiak":GOSUB 1
10
1120 z$="PRINT STRING$(40,208):PRINT"+CH
R$(34)+"Stellen Sie  sich einen Roboter
vor, denSie  mit zahlreichen  Kommandos
 steuernkoennen."+CHR$(34):GOSUB 110
1130 z$="PRINT"+CHR$(34)+"Ich bin dieser
  Roboter,  und ich  werdemich fuer Sie
den Gefahren der verwegen-sten Abenteuer
 aussetzen."+CHR$(34):GOSUB 110
1140 z$="PRINT"+CHR$(34)+"Damit Sie  mic
h sinvoll  agieren  lassenkoennen,  werd
e ich  Ihnen die Situationin der ich  mi
ch gerade befinde, jeweils genau beschre
iben."+CHR$(34):GOSUB 110
1150 z$="PRINT"+CHR$(34)+"Anschliessend
 sagen  Sie  mir  mit zweiWorten,  wie
```

```
beispielsweise  UNTERSUCHETUER, NIMM MES
SER, was ich tun soll."+CHR$(34)+":PRINT
":GOSUB 110
1160 z$="PRINT"+CHR$(34)+"Darueber hinau
s verstehe ich die Befehle      INVENTUR,
 SAVE, LOAD und ENDE."+CHR$(34):GOSUB 11
0
1170 z$="PRINT STRING$(40,210):INK 3,12,
24:INK 2,24,12:PEN 3: PRINT"+CHR$(34)+"
            <Taste>"+CHR$(34)+";:PEN 2:PR
INT"+CHR$(34)+" druecken"+CHR$(34)+";:PE
N 1:PRINT":GOSUB 110
1180 z$="eingabe$=INKEY$:IF eingabe$="+S
TRING$(2,34)+" THEN 980 ELSE CLS:MODE 1:
INK 1,2:INK 2,14:INK 3,26:RETURN":GOSUB
110
1190 t$="Teil III: Adventuretreiber":GOS
UB 160
1200 z$="MODE 1:PAPER 0:INK 0,0:BORDER 0
:INK 1,2:INK 2,14:INK 3,26:PEN 1":GOSUB
110
1210 z$="leerzeile$=STRING$(40,32)":GOSU
B 110
1220 z$="DATA Norden,Sueden,Westen,Osten
,oben,unten":GOSUB 110
1230 z$="FOR richtung=1 to 6":GOSUB 110
1240 z$="READ richtung$(richtung)":GOSUB
 110
1250 z$="NEXT richtung":GOSUB 110
1260 zeilennummer=1070:z$="CLS":GOSUB 11
0
1270 z$="PRINT:PRINT":GOSUB 110
1280 PRINT"- Bildschirmausgabe"
1290 z$="LOCATE 1,1":GOSUB 110
1300 z$="FOR zeile=1 TO 10":GOSUB 110
1310 z$="PRINT leerzeile$;":GOSUB 110
1320 z$="NEXT zeile":GOSUB 110
1330 z$="LOCATE 1,1:PEN 1":GOSUB 110
1340 z$="PRINT"+CHR$(34)+"Ich bin "+CHR$
(34)+";":GOSUB 110
1350 z$="PRINT raum$(spieler);:PRINT"+CH
```

```
R$(34)+"."+CHR$(34):GOSUB 110
1360 z$="PRINT"+CHR$(34)+"Ich sehe "+CHR
$(34)+";:gedruckt=0":GOSUB 110
1370 z$="FOR i=1 TO ao":GOSUB 110
1380 z$="IF ob(i)<>spieler THEN 1210":GO
SUB 110
1390 z$="IF POS(#0)+LEN(ob$(i))+2<=39 TH
EN PRINT ob$(i);"+CHR$(34)+", "+CHR$(34)
+";:GOTO 1205":GOSUB 110
1400 z$="IF POS(#0)+LEN(ob$(i))+2>39 THE
N PRINT:GOTO 1190":GOSUB 110
1410 z$="gedruckt=-1":zeilennummer=1205:
GOSUB 110:zeilennummer=1210
1420 z$="NEXT i":GOSUB 110
1430 z$="IF NOT gedruckt THEN PRINT"+CHR
$(34)+"nichts besonderes  "+CHR$(34)+";"
:zeilennummer=1215:GOSUB 110:zeilennumme
r=1220
1440 z$="PRINT CHR$(8);CHR$(8);"+CHR$(34
)+"."+CHR$(34):GOSUB 110
1450 z$="PRINT leerzeile$:PEN 2":GOSUB 1
10
1460 z$="PRINT"+CHR$(34)+"Ich kann nach
"+CHR$(34)+";:gedruckt=0":GOSUB 110
1470 z$="FOR richtung=1 TO 6":GOSUB 110
1480 z$="IF durchgang(spieler,richtung)=
0 THEN GOTO 1300 ELSE gedruckt=-1":GOSUB
 110
1490 z$="IF POS(#0)=15 THEN PRINT richtu
ng$(richtung);:GOTO 1300":GOSUB 110
1500 z$="IF POS(#0)+LEN(richtung$(richtu
ng))<38 THEN PRINT"+CHR$(34)+", "+CHR$(3
4)+";richtung$(richtung);:GOTO 1300":GOS
UB 110
1510 z$="PRINT"+CHR$(34)+","+CHR$(34)+":
PRINT richtung$(richtung);:GOTO 1300":GO
SUB 110
1520 z$="NEXT richtung":GOSUB 110
1530 z$="IF gedruckt=0 THEN PRINT"+CHR$(
34)+"nirgendwo"+CHR$(34)+";":GOSUB 110
1540 z$="PRINT"+CHR$(34)+"."+CHR$(34)+":
```

```
PEN 3":GOSUB 110
1550 z$="PRINT STRING$(40,154)":GOSUB 11
0
1560 PRINT"- Zugannahme"
1570 zeilennummer=1390:z$="PEN 3:LOCATE
1,25:INPUT"+CHR$(34)+"Was soll ich tun"+
CHR$(34)+";eingabe$:eingabe$=UPPER$(eing
abe$):PEN 2":GOSUB 110
1580 z$="IF LEN(eingabe$)>2 THEN 1500":G
OSUB 110
1590 z$="IF eingabe$="+CHR$(34)+"N"+CHR$
(34)+" AND durchgang(spieler,1)<>0 THEN
spieler=durchgang(spieler,1):PRINT ok$:G
OTO 1080":GOSUB 110
1600 z$="IF eingabe$="+CHR$(34)+"S"+CHR$
(34)+" AND durchgang(spieler,2)<>0 THEN
spieler=durchgang(spieler,2):PRINT ok$:G
OTO 1080":GOSUB 110
1610 z$="IF eingabe$="+CHR$(34)+"W"+CHR$
(34)+" AND durchgang(spieler,3)<>0 THEN
spieler=durchgang(spieler,3):PRINT ok$:G
OTO 1080":GOSUB 110
1620 z$="IF eingabe$="+CHR$(34)+"O"+CHR$
(34)+" AND durchgang(spieler,4)<>0 THEN
spieler=durchgang(spieler,4):PRINT ok$:G
OTO 1080":GOSUB 110
1630 z$="IF eingabe$="+CHR$(34)+"OB"+CHR
$(34)+" AND durchgang(spieler,5)<>0 THEN
 spieler=durchgang(spieler,5):PRINT ok$:
GOTO 1080":GOSUB 110
1640 z$="IF eingabe$="+CHR$(34)+"U"+CHR$
(34)+" AND durchgang(spieler,6)<>0 THEN
spieler=durchgang(spieler,6):PRINT ok$:G
OTO 1080":GOSUB 110
1650 z$="IF LEN(eingabe$)<3 THEN PRINT"+
CHR$(34)+"Dahin fuehrt kein Weg !"+CHR$(
34)+":GOTO 1080":GOSUB 110
1660 z$="IF LEN(eingabe$)>8 THEN GOTO 20
00":GOSUB 110
1670 PRINT"- INVentur"
1680 zeilennummer=1499:z$="REM *********
```

```
******* START INVENTUR":GOSUB 110
1690 zeilennummer=1500:z$="IF LEFT$(eing
abe$,3)<>"+CHR$(34)+"INV"+CHR$(34)+" THE
N GOTO 1600":GOSUB 110
1700 z$="PRINT"+CHR$(34)+"Ich trage folg
endes mit mir:"+CHR$(34):GOSUB 110
1710 z$="FOR objekt=1 TO ao":GOSUB 110
1720 z$="IF ob(objekt)=-1 THEN PRINT ob$
(objekt)":GOSUB 110
1730 z$="NEXT objekt":GOSUB 110
1740 z$="GOTO 1080":GOSUB 110
1750 z$="REM **************** ENDE INVEN
TUR":GOSUB 110
1760 PRINT"- SAVe Game":zeilennummer=160
0
1770 z$="IF LEFT$(eingabe$,3)<>"+CHR$(34
)+"SAV"+CHR$(34)+" THEN GOTO 1700":GOSUB
 110
1780 z$="FOR flag=1 TO af:PRINT#9,fl(fla
g):NEXT flag":GOSUB 70
1790 z$="IF LEN(eingabe$)>10 THEN PRINT"
+CHR$(34)+"Bitte etwas kuerzer !"+CHR$(3
4)+":GOTO 1610 ELSE eingabe$="+CHR$(34)+
"!"+CHR$(34)+"+eingabe$+"+CHR$(34)+".spi
el"+CHR$(34)+":OPENOUT eingabe$":GOSUB 1
10
1800 z$="PRINT#9,spieler":GOSUB 110
1810 z$="FOR objekt=1 TO ao":GOSUB 110
1820 z$="PRINT#9,ob(objekt)":GOSUB 110
1830 z$="NEXT objekt":GOSUB 110
1840 z$="FOR raum=1 TO ar:FOR richtung=1
 TO 6:PRINT#9,durchgang(raum,richtung):N
EXT richtung:NEXT raum":GOSUB 110
1850 z$="FOR flag=1 TO af:PRINT#9,fl(fla
g):NEXT flag":GOSUB 110
1860 z$="CLOSEOUT:PRINT ok$:GOTO 1080":G
OSUB 110
1870 PRINT"- LOAd Game"
1880 z$="IF LEFT$(eingabe$,3)<>"+CHR$(34
```

```
)+"LOA"+CHR$(34)+" THEN GOTO 1900":GOSUB
 110
1890 z$="PRINT"+CHR$(34)+"Cassette rueck
spulen & PLAY druecken ...Welches Spiel
laden ..........."+CHR$(34)+";STRING$(10,
8);:INPUT eingabe$":GOSUB 110
1900 z$="IF LEN(eingabe$)>10 THEN PRINT"
+CHR$(34)+"Das kann nicht sein !"+CHR$(3
4)+":GOTO 1710 ELSE eingabe$="+CHR$(34)+
"!"+CHR$(34)+"+eingabe$+"+CHR$(34)+".SPI
EL"+CHR$(34)+":OPENIN eingabe$":GOSUB 11
0
1910 z$="INPUT#9,spieler":GOSUB 110
1920 z$="FOR objekt=1 TO ao:INPUT#9,ob(o
bjekt):NEXT objekt":GOSUB 110
1930 z$="FOR raum=1 TO ar:FOR richtung=1
 TO 6:INPUT#9,durchgang(raum,richtung):N
EXT richtung:NEXT raum":GOSUB 110
1940 z$="FOR flag=1 TO af:INPUT#9,fl(fla
g):NEXT flag":GOSUB 110
1950 z$="CLOSEIN:PRINT ok$:GOTO 1080":GO
SUB 110
1960 PRINT"- INS":zeilennummer=1900
1970 z$="IF LEFT$(eingabe$,3)<>"+CHR$(34
)+"INS"+CHR$(34)+" THEN GOTO 1950":GOSUB
 110
1980 z$="GOSUB 900:GOTO 1080":GOSUB 110
1990 PRINT"- END":zeilennummer=1950
2000 z$="IF LEFT$(eingabe$,3)<>"+CHR$(34
)+"END"+CHR$(34)+" THEN GOTO 2000":GOSUB
 110
2010 z$="PRINT"+CHR$(34)+"Der Autor wuen
scht Ihnen mehr Erfolg    beim naechsten
 Mal."+CHR$(34)+":PRINT:PRINT:PRINT:END"
:GOSUB 110
2020 PRINT"- Eingabeanalyse":Zeilennumme
r=2000
2030 z$="laenge=LEN(eingabe$)":GOSUB 110
2040 z$="FOR buchstabe=1 TO laenge":GOSU
B 110
2050 z$="pruef$=MID$(eingabe$,buchstabe,
```

```
1)":GOSUB 110
2060 z$="IF pruef$<>"+CHR$(34)+CHR$(32)+
CHR$(34)+"THEN NEXT buchstabe":GOSUB 110
2070 z$="everb$=LEFT$(eingabe$,wl)":GOSU
B 110
2080 z$="rl=laenge-buchstabe":GOSUB 110
2090 z$="IF rl<0 THEN 2090":GOSUB 110
2100 z$="eobjekt$=RIGHT$(eingabe$,rl)":G
OSUB 110
2110 z$="eobjekt$=LEFT$(eobjekt$,wl)":GO
SUB 110
2120 z$="FOR verbnummer=1 TO av":GOSUB 1
10
2130 z$="IF everb$=verb$(verbnummer) THE
N 2130":GOSUB 110
2140 z$="NEXT verbnummer":GOSUB 110
2150 z$="PRINT"+CHR$(34)+"Das Verb verst
ehe ich nicht."+CHR$(34)+":GOTO 1080":GO
SUB 110
2160 z$="FOR o=1 TO ao":GOSUB 110
2170 z$="IF eobjekt$=rufname$(o) THEN 22
00":GOSUB 110
2180 z$="NEXT o":GOSUB 110
2190 z$="PRINT"+CHR$(34)+"Diesen Gegenst
and kenne ich nicht."+CHR$(34)+":GOTO 10
80":GOSUB 110
2200 PRINT"- Sprungtabelle":zeilennummer
=2200:sprungziel=5000
2210 z$="ON verbnummer GOTO 5000"
2220 FOR i=1 TO av
2230 sprungziel=sprungziel+1000
2240 z$=z$+","+STR$(sprungziel)
2250 NEXT i
2260 GOSUB 110
2270 PRINT"- Titel: Spieler tod":zeilenn
ummer=4500
2280 z$="CLS:REM SPIELER TOD":GOSUB 110
2290 z$="PRINT"+CHR$(34)+"Auch das noch
!"+CHR$(34)+":PRINT:PRINT m$(0)":GOSUB 1
10
2300 z$="PRINT:PRINT"+CHR$(34)+"Ich bin
```

```
tod !"+CHR$(34)+":PRINT":GOSUB 110
2310 z$="INPUT"+CHR$(34)+"Soll ich es no
ch einmal versuchen "+CHR$(34)+";eingabe
$:eingabe$=UPPER$(eingabe$)":GOSUB 110
2320 z$="IF LEFT$(eingabe$,1)="+CHR$(34)
+"J"+CHR$(34)+" THEN RUN":GOSUB 110
2330 z$="GOTO 1960":GOSUB 110
2340 PRINT"- Titel: Spieler siegt":zeile
nnummer=4800
2350 z$="CLS:REM SPIELER SIEGT":GOSUB 11
0
2360 z$="PRINT"+CHR$(34)+"Herzlichen Glu
eckwunsch !"+CHR$(34):GOSUB 110
2370 z$="PRINT:PRINT"+CHR$(34)+"Sie habe
n die Ihnen gestellte Aufgabe"+CHR$(34)+
":PRINT:PRINT"+CHR$(34)+"geloest und due
rfen sich nun an einem"+CHR$(34)+":PRINT
:PRINT"+CHR$(34)+"anderen Adventure vers
uchen."+CHR$(34):GOSUB 110
2380 z$="PRINT:PRINT:PRINT:PRINT:END":GO
SUB 110
2390 t$="Teil IV: Bedingungen & Aktionen
":GOSUB 160
2400 zeilenabstand=1:z1=5000:zeilennumme
r=z1:o=0:v=0
2410 v=v+1
2420 o=o+1
2430 x=o:z$="IF o="+STR$(o)+" AND "
2440 GOSUB 160
2450 PRINT" 1 - Objekt ist im Raum"
2460 PRINT" 2 - Objekt ist nicht im Raum
"
2470 PRINT" 3 - Spieler hat Objekt"
2480 PRINT" 4 - Flag ist gesetzt"
2490 PRINT" 5 - Flag ist nicht gesetzt"
2500 PRINT" 6 - Spieler muss sein in Rau
m: ..."
2510 PRINT" 7 - UND weitere Bedingung"
2520 PRINT" 8 - ODER weitere Bedingung"
2530 PRINT:PRINT" 0 - Bedingungen sind O
kay"
```

```
2540 PRINT
2550 PRINT"Waehlen Sie fuer Verb ";v;" O
bjekt ";o
2560 INPUT e
2570 IF e=1 THEN z$=z$+"ob(o)=spieler"
2580 IF e=2 THEN z$=z$+"ob(o)<>spieler"
2590 IF e=3 THEN INPUT"Objekt Nummer ";x
:z$=z$+"ob("+STR$(x)+")=spieler"
2600 IF e=4 THEN INPUT"Flag Nummer   ";x
:z$=z$+"fl("+STR$(x)+")=-1"
2610 IF e=5 THEN INPUT"Flag Nummer   ";x
:z$=z$+"fl("+STR$(x)+")<>-1"
2620 IF e=6 THEN INPUT"Welchen Raum  ";x
:z$=z$+"spieler="+STR$(x)
2630 IF e=7 THEN z$=z$+" AND "
2640 IF e=8 THEN z$=z$+" OR "
2650 IF e<>0 THEN 2440
2660 z$=z$+" THEN "
2670 GOSUB 160
2680 PRINT" 1 - Objekt verschwindet"
2690 PRINT" 2 - Objekt ins Inventory"
2700 PRINT" 3 - Objekt erscheint neu"
2710 PRINT" 4 - Flag wird gesetzt"
2720 PRINT" 5 - Flag wird geloescht"
2730 PRINT" 6 - Durchgang oeffnen"
2740 PRINT" 7 - Mitteilung ausgeben"
2750 PRINT" 8 - Spieler stirbt"
2760 PRINT" 9 - Spieler siegt"
2770 PRINT:PRINT" 0 - Aktionen Okay"
2780 PRINT"Waehlen Sie fuer Verb ";v;" O
bjekt ";o:INPUT e
2790 IF e=1 THEN INPUT"Objekt Nummer ";x
:z$=z$+"ob("+STR$(x)+")=0"
2800 IF e=2 THEN INPUT"Objekt Nummer ";x
:z$=z$+"ob("+STR$(x)+")=-1"
2810 IF e=3 THEN INPUT"Objekt Nummer ";x
:z$=z$+"ob("+STR$(x)+")=spieler"
2820 IF e=4 THEN INPUT"Flag Nummer ";x:z
$=z$+"fl("+STR$(x)+")=-1"
2830 IF e=5 THEN INPUT"Flag Nummer ";x:z
$=z$+"fl("+STR$(x)+")=0"
```

```
2840 IF e=6 THEN INPUT"von Raum ";x:INPU
T" in Richtung ";y:INPUT"  nach Raum ";z
:z$=z$+"durchgang("+STR$(x)+","+STR$(y)+
")="+STR$(z)
2850 IF e=7 THEN INPUT"und zwar:
                              ";eingabe$:z$=z$+"
PRINT"+CHR$(34)+eingabe$+CHR$(34)
2860 IF e=8 THEN INPUT"Mitteilung fuer d
en Spieler eingeben   ";eingabe$:z$=z$+"
m$(0)="+CHR$(34)+eingabe$+CHR$(34)+":GOT
O 4500"
2870 IF e=9 THEN z$=z$+"GOTO 4800"
2880 IF e=0 THEN z$=z$+"GOTO 1080":GOSUB
 110:ELSE z$=z$+":":GOTO 2670
2890 INPUT"Mehr zum gleichen Objekt ";ei
ngabe$:eingabe$=UPPER$(eingabe$)
2900 IF LEFT$(eingabe$,1)="J" THEN 2410
2910 IF o<>ao THEN GOTO 2420
2920 o=0:z1=z1+1000:zeilennummer=z1
2930 IF v<>av THEN GOTO 2410
2940 CLOSEOUT
2950 CLS
2960 PRINT"Auf dem verwendeten Datentrae
ger":PRINT:PRINT"existiert nun ein Progr
amm mit Namen":PRINT:PRINT MID$(adventur
e$,2);"  welches Sie auf uebliche":PRINT
:PRINT"Weise laden koennen.":PRINT:PRINT
:PRINT:PRINT"Viel Spass !":PRINT:PRINT:E
ND
```

GRAFIK PROGRAMMER

Beschreibung

GRAFIK - PROGRAMMER

Das Ihnen im folgenden als Listing vorgestellte Programm wurde von mir entwickelt, um einfache HIRES - Grafiken auf leichte und schnelle Art programmieren zu können.

Die Vorgehensweise ist denkbar einfach: Sie erzeugen unter Verwendung eines tastengesteuerten Cursors und mit Hilfe einiger vom Editor gestellter Fragen die Grafik auf dem Bildschirm. Die für die Erzeugung dieses Bildes notwendigen Daten werden gespeichert und später zur Erstellung eines Programmes verwendet, welches als Unterprogramm an andere Programme angefügt werden kann.
Somit ist es nicht erforderlich, daß Sie die jeweiligen Koordinaten mit irgendwelchen Hilfsmitteln, wie beispielsweise Bildschirmarbeitsblättern, ermitteln müssen.

ERSTELLUNG VON GRAFIKEN MIT GRAFPRO

Nach dem Titelbild zeigt der Editor Ihnen zunächst das Hilfsmenue, welches Sie jederzeit auch durch Betätigung der Taste H erreichen können.
Sobald Sie sich über die Ihnen zur Verfügung stehenden Möglichkeiten informiert haben, können Sie durch drücken einer beliebigen Taste in den Editiermodus umschalten.

Es stehen Ihnen nun eine Reihe von Zeichen- wie auch Steuerbefehlen zur Verfügung.
Zeichenbefehle werden mittels Klein-, Steuerbefehle mittels Großbuchstaben angesprochen.

CURSORBEWEGUNG:

Der Cursor ist ein nicht zerstörender, schnell blinkender Punkt auf dem Grafikschirm. Zu seiner Steuerung dienen die Tasten des Zehnerblocks:

7 8 9
4 5 6
1 2 3

wobei deren Anordnung auf der Tastatur der jeweiligen Bewegungsrichtung entspricht.
Ein Druck auf die fünf wird den Cursor in der Bildschirmmitte positionieren.
Bei Programmstart befindet er sich immer auf der Position 0,0, somit in der linken unteren Ecke.
Damit die Ansteuerung eines beliebigen Punktes auf dem Bildschirm nun nicht zu einer Geduldsarbeit wird, haben Sie die Möglichkeit, die Schrittweite des Cursors nach Eingabe von *C* beliebig groß festzulegen.

ZEICHNEN

Um einen Punkt zu zeichnen, steuern Sie den Cursor auf die gewünschte Position, und drücken *P* für Punkt.
Sofort wird der Punkt gesetzt, die entsprechenden Koordinaten werden für die spätere Programmierung gespeichert.

Wünschen Sie eine Linie zu zeichnen, so betätigen Sie die *L*-Taste, und sie wird ausgehend von der zuletzt gesetzten Koordinate bis zur Position des Cursors gezeichnet.
Dabei ist es gleichgültig, ob der letzte Punkt mit *P* gesetzt wurde, oder ob es sich um den Endpunkt einer zuvor gezeichneten Linie handelt.

Z rekonstruiert jederzeit (z.B. nach dem Laden) ein im Speicher befindliches Bild.

FEHLERBEHANDLUNG

Sollte Ihnen während des Zeichnens ein Fehler unterlaufen sein, so ist Ihr Werk noch nicht verloren.
Nach Eingabe von wird der Bildschirm gelöscht und das Bild neu gezeichet, allerdings ohne die zuletzt gezeichnete Linie und ohne den zuletzt gesetzten Punkt. Diesen Vorgang können Sie beliebig oft wiederholen.

BILD SPEICHERN, BILD LADEN

Betätigen Sie ***L*** oder ***S*** und geben Sie einen bis zu sechzehn Buchstaben langen Namen für das Bild ein. Grafpro fügt nach Betätigung von ENTER den gewünschten Vorgang aus.

BILD PROGRAMMIEREN

Nach Niederhalten von ***P*** ist wiederum die Eingabe eines Namens erforderlich. Anschließend werden Sie nach der Zeilennummer gefragt, mit der das Programm beginnen soll, und es wird ein entsprechendes Programm erzeugt.

VERWENDUNG DER ERZEUGTEN PROGRAMME

Die generierten Programme wurden als ASCII - Datei auf dem verwendeten Datenträger abgelegt und können an bestehende Programme angefügt werden.
Beachten Sie bitte, daß dann jedoch der Ladevorgang durch MERGE eingeleitet werden muß, außerdem sollte keine Zeile in beiden Programmen belegt sein, da sonst die Zeilennummern des Hauptprogrammes während des Ladevorganges überschrieben werden.

Wird das Bildprogramm mit LOAD geladen, so wird der bisherige Speicherinhalt gelöscht.

GRAFIK PROGRAMMER

Listing

GRAFIK PROGRAMMER

Listing

```
1 REM ********************************
2 REM * Grafik Programmer; Version 1m *
3 REM * (c) 11.84  by Joerg Walkowiak *
4 REM ********************************
5 CLS
30 DEFINT a-z
40 DIM x(200),y(200),modus$(200)
50 c=10:farbe=1
120 LOCATE 4,5:PRINT"Grafik - Programmer
 GRAFPRO CPC 1m"
130 LOCATE 13,6:PRINT"by J";CHR$(178);"r
g Walkowiak"
135 LOCATE 1,12:PRINT STRING$(40,154)
140 LOCATE 10,10:PRINT CHR$(164);" 1984
by DATA BECKER"
150 LOCATE 9,14:PEN 2:PRINT"Aus dem DATA
 BECKER Buch":LOCATE 16,16:PRINT"Adventu
res":PRINT"    und wie man sie auf dem S
chneider":LOCATE 13,18:PRINT"CPC program
miert.":PEN 1
190 FOR i=1 TO 20000:NEXT i:GOTO 2000
1000 eingabe$=INKEY$
1020 IF eingabe$="8" THEN Y=Y+C:GOTO 190
0
1030 IF eingabe$="6"THEN X=X+C:GOTO 1900
1040 IF eingabe$="4"THEN X=X-C:GOTO 1900
1050 IF eingabe$="2"THEN Y=Y-C:GOTO 1900
1100 IF eingabe$="p" THEN x(p)=x:y(p)=y:
modus$(p)="p":PLOT x,y,farbe:p=p+1:lx=x:
ly=y:GOTO 1000
1105 IF eingabe$="l" THEN  modus$(p)="l"
:x(p)=x:y(p)=y:PLOT lx,ly,farbe:DRAW x,y
,farbe:lx=x:ly=y:p=p+1:GOTO 1000

1110 IF eingabe$="P" THEN GOTO 3010
1120 IF eingabe$="z" THEN GOTO 6010
1130 IF eingabe$="S" THEN 4010
1150 IF eingabe$="L" THEN 5010
1160 IF eingabe$="9"THEN x=x+c:y=y+c:GOT
O 1900
1170 IF eingabe$="7"THEN x=x-c:y=y+c:GOT
```

```
O 1900
1180 IF eingabe$="1"THEN x=x-c:y=y-c:GOT
O 1900
1190 IF eingabe$="3"THEN x=x+c:y=y-c:GOT
O 1900
1200 IF (eingabe$="h" OR eingabe$="H") T
HEN GOTO 2000
1210 IF eingabe$="5"THEN x=320:y=200:GOT
O 1900
1220 IF eingabe$="c" THEN CLS:INPUT"neue
 Cursorschrittweite";c:GOTO 6010
1230 IF eingabe$="@"THEN p=p-1:GOTO 6010
1900 farbe1=TEST(x,y)
1910 FOR x1=1 TO 10
1920 PLOT x,y,farbe1+1
1930 PLOT x,y,farbe1
1940 NEXT x1
1950 GOTO 1000
1999 REM ************************** inst
2000 MODE 1:CLS:LOCATE 15,1:PRINT"  Hilf
smenue":PRINT STRING$(40,154);:PRINT
2010 PRINT"7  8  9           ";:PEN 2:PRIN
T"Cursorbewegung":PEN 1
2020 PRINT"4     6           entsprechend
der
2030 PRINT"1  2  3           Anordnung der
 Tasten                      auf dem 12-er
 Block"
2040 PRINT"   5              Cursor auf Sc
hirmmitte":PRINT
2050 PEN 2: PRINT"Zeichenbefehle";:PEN 1
:PRINT" mit Kleinbuchstaben:"
2060 PRINT" p - Punkt an Cursorposition
setzen"
2070 PRINT" l - Linie vom zuletzt gesetz
ten Punkt,       oder vom Endpunkt einer
Linie aus,       zeichnen"
2080 PRINT" z - Bild neu zeichnen"
2090 PRINT" c - Cursorschrittweite setze
n"
2100 PRINT
```

```
2110 PEN 2:PRINT"Steuerbefehle";:PEN 1:P
RINT" mit Grossbuchstaben:"
2120 PRINT" H - Hilfsmenue
2130 PRINT" S - Bilddaten speichern
2140 PRINT" L - Bilddaten laden
2150 PRINT" P - Basicprogramm erzeugen"
2160 PEN 2:PRINT" @ - macht letzten Befe
hl unwirksam":PEN 1
2170 PRINT STRING$(40,154);:PRINT"Wenn O
.K. dann beliebige Taste druecken.";

2180 IF INKEY$="" THEN 2180
2190 CLS:x=320:y=200:GOTO 1000
3000 REM ************************ prog
3010 CLS:PRINT"Grafik programmieren ..."
:PRINT STRING$(40,154)
3020 INPUT"Wie soll das Programm heissen
";name$:IF LEN(name$)>16 THEN GOTO 3010

3030 IF name$="@"THEN GOTO 6010
3040 PRINT:INPUT"Nummer der ersten Progr
ammzeile";zeile
3050 PRINT:INPUT"Welchen Zeilenabstand w
uenschen Sie";zeilenabstand:PRINT
3060 WINDOW #1,1,40,18,25
3070 OPENOUT name$
3080 FOR i=0 TO p-1
3090 IF modus$(i)="p"THEN befehl$="plot"
:GOTO 3110
3100 IF modus$(i)="l"THEN befehl$="draw"

3110 zeile$=befehl$+STR$(x(i))+","+STR$(
y(i))
3120 IF LEN(progzeile$)<2 THEN progzeile
$=zeile$:zeile$="":GOTO 3160
3130 IF (LEN(progzeile$)<200 AND LEN(pro
gzeile$)>5) THEN progzeile$=progzeile$+"
:"+zeile$:zeile$="":GOTO 3160
3140 PEN 3:PRINT#9,zeile;progzeile$:PEN
2:PRINT#1,zeile;progzeile$
3150 zeile=zeile+zeilenabstand:progzeile
```

```
$=zeile$
3160 NEXT i
3170 IF LEN(progzeile$)>1 THEN PEN 3:PRI
NT#9,zeile;progzeile$:PEN 2:PRINT#1,zeil
e;progzeile$
3180 PEN 3:CLOSEOUT
3190 PEN 1
3200 GOTO 6010
4000 REM ************************** save
4010 CLS:PRINT"Bilddaten speichern ...":
PRINT STRING$(40,"_")
4020 INPUT"Wie soll das Bild heissen";na
me$:IF LEN(name$)>16 THEN GOTO 4010
4030 IF name$="@"THEN GOTO 6010
4040 OPENOUT name$
4050 PEN 3:PRINT
4060 PRINT#9,p
4070 FOR i=0 TO p
4080 PRINT #9,x(i),y(i),modus$(i)
4090 NEXT i
4100 CLOSEOUT
4110 PEN 1
4120 GOTO 6010
5000 REM ************************** load
5010 CLS:PRINT"Bilddaten laden ...":PRIN
T STRING$(40,"_")
5020 INPUT"Welches Bild";name$:IF LEN(na
me$)>16 THEN GOTO 5010
5030 IF name$="@"THEN GOTO 6010
5040 PRINT:PEN 2
5050 OPENIN name$
5060 INPUT#9,p
5070 FOR i=0 TO p-1
5080 INPUT#9,x(i),y(i),modus$(i)
5090 NEXT i
5100 CLOSEIN
5110 PEN 1
6000 REM ************************** zeich
6010 CLS:FOR i=0 TO p-1
6020 IF modus$(i)="p" THEN PLOT x(i),y(i
),farbe:GOTO 6040
```

```
6030 IF modus$(i)="1" THEN DRAW x(i),y(i
),farbe
6040 NEXT i
6050 GOTO 1000
```

7. ANHANG

PROGRAMMVERBESSERUNGEN

Die in diesem Buch abgedruckten Programme sollen Ihnen Anregungen und Beispiele für Ihre eigene Arbeit liefern, daß heißt sie müssen überschaubar bleiben damit Sie sich in ihnen ohne Mühen zurechtfinden können.
Leider stellen aber für uns leicht verständliche Programme, zumindest solange sie in Basic geschrieben sind, keineswegs ein Optimum für den Computer dar.
Aus diesem Grunde wird es sich sicherlich auch für Sie lohnen, die vorgestellte Software unter Berücksichtigung der folgenden Aspekte noch einmal zu überarbeiten.

Natürlich gilt das hier Gesagte nicht nur für unsere Adventures, sondern für alle BASIC - Programme !

SPEICHERPLATZOPTIMIERUNG

Achten Sie darauf, die einzelnen Programmzeilen möglichst zu füllen; verwenden Sie somit die geringstmögliche Anzahl von Zeilen.
Denn jede einzelne Anweisungszeile belegt mehrere Bytes für die eigene Zeilennummer, zusätzlich müssen Daten zur Auffindung der nächsten Programmzeile abgelegt werden.

Verzichten Sie auf alle REMarks !

Benutzen Sie statt der von uns zur Erhöhung der Lesbarkeit verwandten langen Variablennamen nur ein oder zwei Buchstaben lange Bezeichnungen !

ABLAUFOPTIMIERUNG

In der professionellen Datenverarbeitung aus Kostengründen, in unserem Falle wegen der besseren Spielbarkeit, ergibt sich ein Streben nach möglichst schnell ablaufenden Programmen.
Der einzige Weg zu diesem Ziel zwingt uns, auf die Belange des Computers und dessen Betriebssystem Rücksicht zu nehmen.

Folgende Vorschläge helfen dem Basic - Interpreter, Ihre Programme möglichst schnell ablaufen zu lassen. Natürlich tragen auch die zuvor gemachten Vorschläge zu einem Speed - Up bei.

Dieser erstellt während seiner Arbeit eine Reihe von Listen, wobei die Reihenfolge der einzelnen Elemente von der Reihenfolge ihres Eintragens in die jeweilige Liste abhängig ist.
So wird jeder Variablen in der Reihenfolge ihres Auftretens Platz in dem dafür vorgesehenem Abschnitt des Programmspeichers zugewiesen.
Greift der Rechner nun zum Lesen oder Schreiben auf eine bestimmte Variable zu, so muß er alle Elemente dieser Liste durchgehen, bis die geforderte Variable erreicht worden ist.
Deshalb ist es zweckmäßig, die Variablen in der Reihenfolge ihrer Häufigkeit eintragen zu lassen, was durch eine Programmzeile, in der diese Variablen beispielsweise auf Null gesetzt werden, geschehen kann.

Außerdem sollten Sie auch häufig benutzte Konstanten zuvor als Variablen definieren, denn diese festen Zahlen müssen bei jeder einzelnen Operation auf das Verarbeitungsformat des Interpreters umgerechnet werden, was ansonsten nur einmal bei der Definition geschieht.

Berücksichtigen Sie diesen Tip insbesondere bei Schleifenkonstruktionen !

Zum Verarbeitungsformat gilt es weiterhin zu sagen, daß ganze Zahlen mit Sicherheit schneller zu verarbeiten sind als Gleitkommazahlen.
Aus diesem Grunde sollten Sie jedes Programm mit einem DEFINT a-z beginnen.
Denn fast jeder Programmierer fügt an Stringvariablen sowieso ein Komma an, und die Anzahl der wirklich erforderlichen Variablen mit höherer Genauigkeit wird in fast jedem Programm geringer als die der übrigen sein.

Unser zweiter Ansatzpunkt betrifft die Sprünge, die ein Programm ausführt. Denn woher weiß der Interpreter, an welcher Stelle im Speicher sich das Sprungziel befindet ?
Ganz einfach, er weiß es nicht, sondern er geht alle Zeilen durch, bis er die vorgegebene Zeile erreicht hat.
Legen Sie daher alle häufig benutzten Subroutinen an den Beginn Ihrer Programme, um die Suchzeit, auch wenn es sich nur um winzige Bruchteile von Sekunden handelt, zu verkürzen !

STICHWORTVERZEICHNIS

DAS STEHT DRIN:

Ein faszinierender Führer in die phantastische Welt der Abenteuer-Spiele. Hier wird gezeigt wie Adventures funktionieren, wie man sie erfolgreich spielt und wie man eigene Adventures auf dem CPC 464 programmiert. Dabei wird das gesamte Spektrum bis hin zum trickreichen Grafikadventure abgehandelt und mit vielen Programmbeispielen belegt.

Der Clou des Buches ist, neben vielen fertigen Adventures zum Abtippen, ein kompletter ADVENTURE-GENERATOR, mit dem das Selberprogrammieren packender Adventures zum Kinderspiel wird.

Zusätzlich wird ein Grafik-Editor und -Programmierer vorgestellt, mit dessen Hilfe Sie im Handumdrehen die für Ihre Grafikadventures notwendigen Unterprogramme erzeugen können.

UND GESCHRIEBEN HAT DIESES BUCH:

Jörg Walkowiak, Informatik-Student, der auf jahrelange Erfahrung mit den gängigsten Heim- und Personalcomputern zurückblicken kann und erfolgreicher Autor professioneller Adventures ist.

ISBN 3-89011-088-6

Walkowiak / Adventures – und wie man sie auf dem CPC 464 programmiert

